VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra telekomunikační techniky

Software Factory- Komponenta pro Wi-Fi a Bluetooth ad-hoc síť

Software Factory – Wi-Fi and Bluetooth ad-hoc Network Components

2013 **Daniel Pietrosz**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Daniel Pietrosz**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Software Factory - Komponenta pro WIFI a bluetooth adhoc síť
Software Factory - WIFI and Bluetooth ad-hoc Network Components

Zásady pro vypracování:

Cílem této diplomové práce je podílet se na činnostech a rozvoji laboratoře - Software Factory na katedře Informatiky FEI. Dlouhodobým cílem této laboratoře je vytvořit a rozvíjet prostředí pro výuku, výzkum a vývoj softwarového inženýrství, díky kterému bude možné simulovat a sledovat reálný vývoj aplikací, testovat různé procesy a postupy apod.
Konkrétním cílem této diplomové práce je zaměřit se na vývoj aplikací pro mobilní telefony Android, vyvinout softwarové komponenty pro poloautomatické (automatické) nastavení WIFI adhoc sítě a bluetooth sítě.
1. Rešerše současného stavu používaných modulů WIFI a bluetooth v mobilech s OS Android. Rozhraní pro práci s těmito moduly.
2. Návrh a implementace komponenty pro WIFI adhoc síť.
3. Návrh a implementace komponenty pro bluetooth adhoc síť.
4. Otestování obou komponent v reálné aplikaci.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * FAKULTA elektrotechniky a informatiky * OSTRAVA

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 6. 5. 2013 .....

Podpis autora

Poděkování

Děkuji svému vedoucímu Ing. Svatoplukovi Štolfovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce. Děkuji i svým rodičům za psychickou podporu při realizaci práce.

Abstrakt

Tato diplomová práce se zabývá vývojem komponent Wi-Fi a Bluetooth pro OS Android. Tyto komponenty mají sloužit vývojářům k propojení a komunikaci více zařízení pomocí Wi-Fi nebo Bluetooth. Práce obsahuje také popis použitých technologií, rešerši současného stavu používaných modulů Wi-Fi a Blueooth v zařízeních s OS Android. Dále práce obsahuje návrh, implementaci a otestování obou komponent.

Klíčová slova

OS Android, Wi-Fi, Bluetooth, komponenta, Sockets, BluetoothSockets

Abstract

This Thesis is about development Wi-Fi and Bluetooth Components for OS Android. Components should serve to connection and communication more Equipments by Wi-Fi or Bluetooth. Content of Thesis has Description of used technologies, Search of current used Moduls Wi-Fi and Bluetooth in OS Android Equipment and Proposal of implemention and test both Components.

Keywords

OS Android, Wi-Fi, Bluetooth, component, Sockets, BluetoothSockets

Seznam použitých symbolů a zkratek

A2DP	Advanced Audio Distribution Profile
ACL	Asynchronous Connection-Less
AIDL	Android Interface Definition Language
AP	Access Point
API	Application Programming Interface
DPSK	Differential Phase Shift Keying
DSSS	Direct Sequence Spread Spectrum
EDR	Enhanced Data Rate
FHSS	Frequency Hopping Spread Spectrum
FM	Frequency Modulation
GFSK	Gaussian Frequency-Shift Keying
GPRS	General Packet Radio Service
GPS	Global Positioning System
HTTP	Hyper Text Transport Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
iOS	iPhone Operating System
IP	Internet Protocol
IPC	Inter-process communication
ISM	Industrial, Scientific, Medical
Java ME	Java 2 Micro Edition
Java SE	Java 2 Standard Edition
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LAN	Local Area Network
MAC	Media Access Control
MIMO	Multiple-input multiple-output
NFC	Near field communication
OFDM	Orthogonal frequency-division multiplexing
OpenGL	Open Graphics Library
OS	Operating system
OS	Operating system
PAN	Personal Area Network
PPP	Point-to-Point Protocol
QoS	Quality of services
SCO	Synchronous Connection Orientated
SDIO	Secure Digital Input Output

SDL	Simple DirectMedia Layer
SIG	Special Interest Group
SSID	Service Set Identifier
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User interface
UML	Unified Modelling Language
UUID	Universally unique identifier
VDP	Video Distribution Profile
WAE	Wireless Audio Experience
WAP	Wireless Application Protocol
WEP	Wireless Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access version 2
XML	Extensible Markup Language
YAML	Yet Another Markup Language

Obsah

1. Úvod

V dnešní době zažíváme obrovský technologický pokrok, především v oblasti chytrých telefonů a tabletů. O tyto zařízení je každým rokem větší zájem. Prodeje lámou rekordy a výrobci mají nebývalý zisk. Tím, jak narůstá počet uživatelů těchto zařízení, roste i poptávka po obsahu do nich. Proto se, čím dál tím víc vývojářů zabývá vývojem aplikací a her právě pro tyto zařízení. Zároveň velká část mobilních aplikací a her je soustředěna na komunikaci a hraní přes internet a jen malá část umožňuje komunikaci více zařízení přes Bluetooth nebo v lokální síti přes Wi-Fi.

Pokud vývojář vyvíjí nějakou aplikaci nebo hru, která bude umožňovat komunikaci a hraní přes internet s největší pravděpodobností sáhne po některém síťovém enginu, který mu v rámci svých služeb poskytne i server. Jestliže však chce umožnit své aplikaci nebo hře komunikaci přes Bluetooth nebo v lokální síti přes Wi-Fi neexistuje komponenta, která by mu vývoj usnadnila. Tyto poznatky mě inspirovaly při výběru tématu diplomové práce.

Při rozhodování pro jaký mobilní operační systém mají být komponenty vyvíjeny bylo uvažováno nad dvěmi. A to nad OS Android od Googlu a iOS od Applu. Oba tyto mobilní OS jsou dnes v chytrých telefonech a tabletech nejrozšířenější. V roce 2012 vlastnili dohromady OS Android a iOS 87,6% podíl celosvětového trhu s mobilními operačními systémy. Konkrétně OS Android 68,8% a iOS 18,8% [41]. Z důvodu většího podílu na trhu byl zvolen OS Android.

Cílem této práce je vytvoření Bluetooth a Wi-Fi komponent, které umožní spojení a komunikaci více zařízení přes Bluetooth a Wi-Fi. Kromě toho je cílem také poskytnout vývojářům základ ke komunikaci a hraní na více zařízeních přes Bluetooth nebo v lokální síti přes Wi-Fi pro jejich aplikace nebo hry. Tyto komponenty jsou následně uplatněny v praxi a to ve hře vytvořené pro OS Android.

Samotná práce je rozdělena do devíti částí, z nichž první tvoří úvod. Druhá část se zabývá popisem použitých technologií, které byly uplatněny při vývoji komponent. Následující část se věnuje rešerši současného stavu a používaných modulů Wi-Fi a Bluetooth v zařízeních s OS Android, která má za úkol seznámení s nejčastěji osazovanými moduly a čipy v zařízeních s OS Android. Ve čtvrté části jsou popsáné rozhraní pro práci s Bluetooth a Wi-Fi v OS Android. Seznámení s multiplayer řešeními pro OS Android je v páté části. V šesté a sedmé části je popsána analýza, návrh a implementace obou komponent. Osmá část se věnuje samotné realizaci obou komponent v reálné aplikaci. Poslední část tvoří závěr, ve kterém jsou zhodnoceny dosažené výsledky a možná vylepšení do budoucna.

2. Použité technologie

2.1 Android

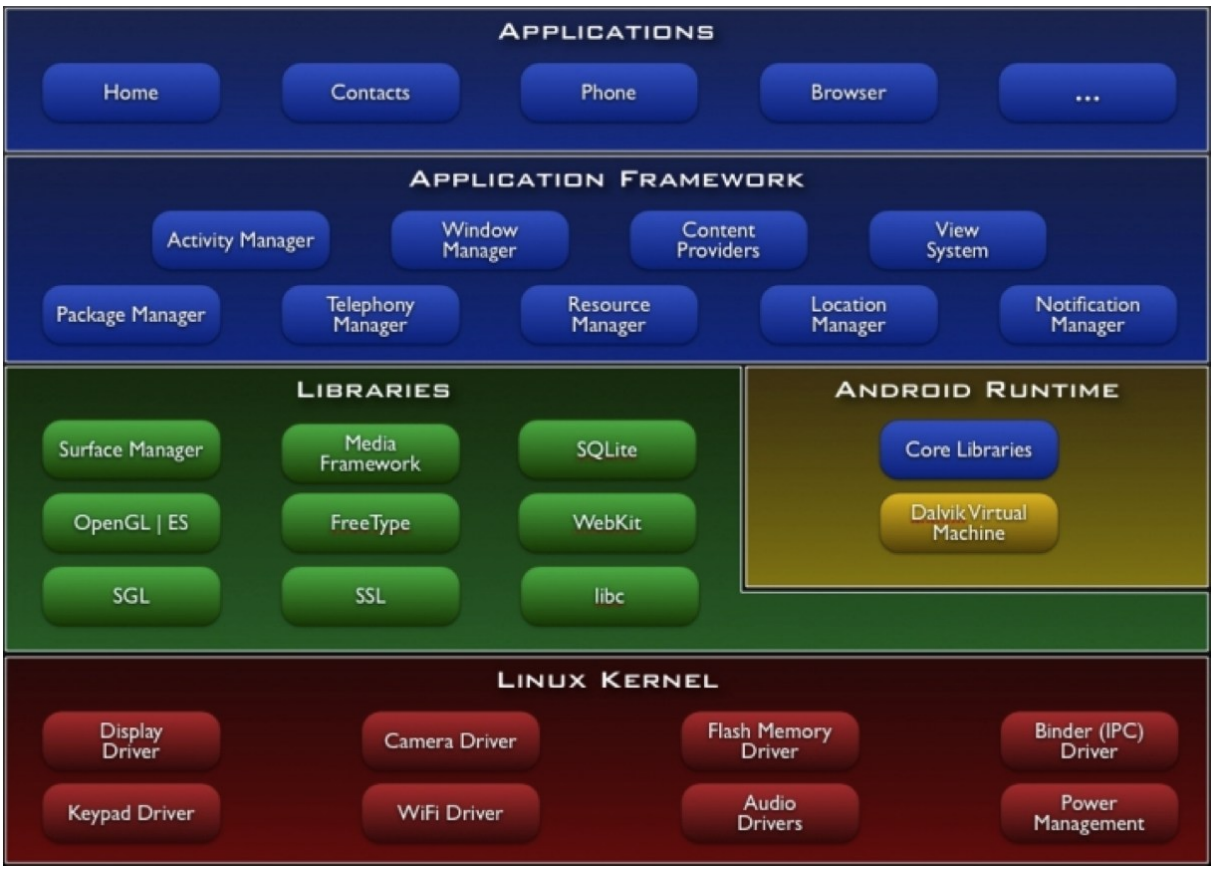
Android je open source mobilní operační systém, který je založen na modifikovaném jádru Linuxu. Původně byl vyvíjen společností stejného jména, Android Inc., kterou v roce 2005 koupil Google, jako součást své strategie pro vstup na trh „chytrých“ mobilních telefonů.
Dne 5. listopadu 2007 bylo vytvořeno uskupení Open Handset Alliance skládající se z několika desítek společností – včetně Googlu, jedná se především o společnosti vyrábějící mobilní telefony, polovodičové součástky, ale mimo jiné, i mobilní operátory.

Cílem tohoto konsorcia je spravovat a vyvíjet otevřený standard pro mobilní zařízení, ale taktéž má za úkol kooperovat vývoj tohoto operačního systému.

V roce 2008 byl na trh uveden první, komerčně dostupný mobilní telefon běžící na systému android T-Mobile G1(HTC Dream). Od roku 2008, měl Android řadu aktualizací, které postupně zlepšily operační systém, přidaly mnoho nových funkcí a opravují chyby ze starších verzí. Aktuálně se Android nachází ve verzi 4.2.2 Jelly Bean. [1,2,3]

2.1.1 Architektura

Architektura OS Android, je rozdělena do několika vrstev a každá využívá služeb, které jsou poskytovány vrstvou pod ní. Následující diagram (Obrázek č. 1),ukazuje hlavní součásti OS Android. Každá sekce je podrobněji popsána níže.



Obrázek č. 1 – Architektura systému Android, zdroj [4]

Vrstvy [1,3,4]:

Linux Kernel

Základní vrstvu tvoří Linux Kernel ve verzi 2.6.x pro Android verze 1,2,3 a Linux Kernel 3.0. x od Androidu 4, který komunikuje s hardwarem a obsahuje všechny základní ovladače hardwaru. Android používá toto jádro pro všechny své základní funkce, jako jsou správa paměti, správa procesů, síťové spojení, nastavení zabezpečení apod.

Libraries

Další vrstvu tvoří na Androidu nativní knihovny. Je to vrstva, která umožňuje, aby zařízení mohlo zpracovávat různé typy dat. Tyto knihovny jsou psány v C nebo C + + jazyku a jsou specifické pro konkrétní hardware. Tyto knihovny nemohou fungovat samostatně. Existují proto, aby byly volány z vyšších vrstev.

Některé z důležitých nativních knihoven jsou:

- **Surface Manager** - stará se o zobrazování aplikací a jejich vrstvení.

- **Media Framework** - poskytuje různé multimediální kodeky, které umožňují nahrávání a přehrávání různých mediálních formátů.

- **SQLite** - je databázový engine použitý v Androidu pro ukládání a práci s daty.

- **WebKit** – open source vykreslovací jádro pro webový prohlížeč.

- **OpenGL** - používá se k vykreslování 2D nebo 3D grafiky na obrazovce.

Android Runtime

- **Dalvik Virtual Machine** - je to typ JVM používaný v zařízeních Android, ke spuštění aplikace a je optimalizován pro zařízení s nízkým výpočetním výkonem a s malou pamětí. Na rozdíl od JVM neběží v Dalvik Virtual Machine soubory .class, místo toho v něm běží .dex soubory. Soubory .dex jsou vytvořeny ze souborů .class v době kompilace a poskytují vyšší účinnost. Dalvik Virtual Machine umožňuje více instancí virtuálního stroje, které mohou být vytvořeny současně. Dále poskytuje bezpečnost, izolaci, správu paměti a podporu vláken.

- **Core Java Libraries** - ty se liší od Java SE a Java ME knihoven. Nicméně tyto knihovny poskytují většinu funkcí definovaných v knihovnách Java SE.

Aplication Framework

Jedná se o bloky, s kterými naše aplikace komunikuje přímo. Tyto programy spravují základní funkce telefonu, jako je řízení zdrojů, správa hovoru atd., které umožňují programátorovi pracovat s prvky operačního systému

Důležité bloky aplikačního rámce jsou:

- Activity Manager - řídí činnost životního cyklu aplikací.
- Content Providers - Správa sdílení dat mezi aplikacemi.
- Notication Manager - umožňuje zobrazovat aplikacím upozornění ve stavovém řádku.
- Resource Manager - podporuje přístup k externím zdrojům např. obrázky, zvuk, atd.
- View System - obsahuje komponenty, které se používají při konstrukci uživatelského rozhraní aplikace. Např. RelativeLayout, Textview, Button, atd.

Aplication

Aplikace se nacházejí ve vrchní vrstvě architektury Android. Nacházejí se zde standardní aplikace, které jsou již přeinstalovány se systémem jako např. SMS klient, telefon, webový prohlížeč, diář, fotoaparát apod. Kromě těchto aplikací jsou zde i aplikace třetích stran, které se nacházejí ve stejné architektonické vrstvě a využívají stejné API knihovny jako aplikace dodávané se systémem.

2.1.2 Hlavní stavební bloky

Hlavní stavební bloky jsou komponenty, které používají vývojáři k vytvoření aplikací pro Android. Každá komponenta má jiný bod, jehož prostřednictvím může aplikace a uživatel zadávat systému své žádosti. Ne všechny komponenty jsou skutečné vstupní body pro uživatele. Některé komponenty jsou závislé na jiné komponentě, ale každá z nich existuje jako svá vlastní entita, která hraje specifickou roli – tj. každá z nich je unikátní stavební blok, který pomáhá definovat celkové chování aplikace. Každý typ slouží k odlišnému účelu a má odlišný cyklus, který definuje jak je komponenta vytvořena a zničena.

V současnosti existují čtyři různé komponenty [3,5]:

Actvity

Aktivita je obvykle jednotlivá obrazovka, kterou vidí uživatel na zařízení. Aplikace má typicky několik aktivit a uživatel prochází mezi nimi tam i zpět. Jako takové, jsou aktivity nejvíce viditelnou částí aplikace. Například, e-mailová aplikace může mít jednu aktivitu, která zobrazuje seznam nových e-mailů, další aktivitu k napsání e-mailu a další aktivitu pro čtení e-mailů. Ačkoli aktivity spolupracují, aby tvořily kompletní e-mailovou aplikaci, každá z nich je nezávislá na ostatních. To přináší mnoho výhod. Třeba kamerová aplikace může začít aktivitu v e-mailové aplikaci, která tvoři novy e-mail, aby mohl uživatel sdílet obrázek, aniž by došlo k ukončení aplikace.

Service

Služby běží na pozadí a nemají žádné komponenty uživatelského rozhraní. Mohou vykonávat stejné akce jako aktivity, ale bez jakéhokoliv uživatelského rozhraní. Služby jsou užitečné pro akce, které chceme vykonávat bez ohledu na tom co se děje na obrazovce. Příkladem může být služba - přehrávat hudbu na pozadí, zatímco uživatel se nachází v jiné aplikaci, nebo může načítat data přes síť, aniž by byla blokována interakce uživatele s aktivitou. Služby mají mnohem jednodušší životní cyklus než aktivity. Lze je spustit nebo zastavit. Také životní cyklus je více méně řízen vývojářem a ne tolik systémem.

V důsledku toho musí mít vývojáři na paměti, aby spouštěli služby tak, aby sdílené prostředky nevyužívaly zbytečně např. CPU, baterii.

Content providers

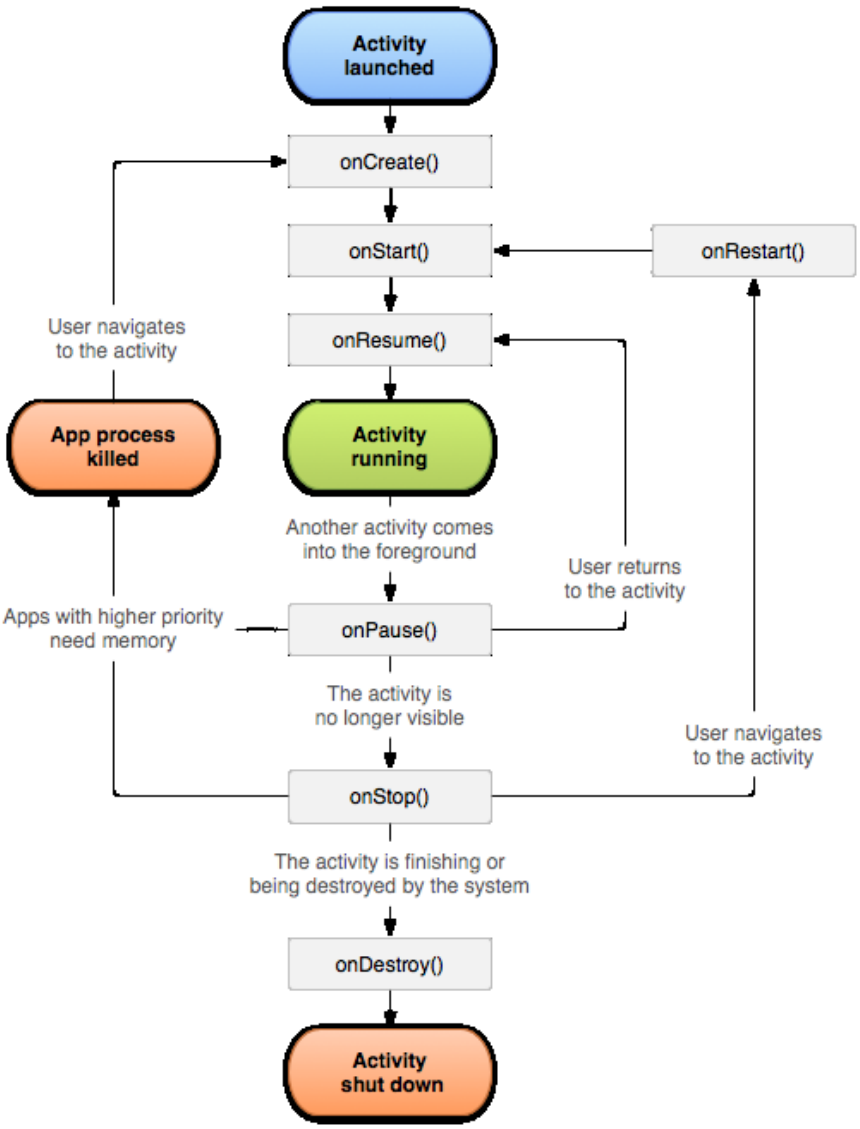
Content providers jsou rozhraní pro sdílení dat mezi aplikacemi, ale taktéž slouží pro sdílení dat v aplikaci mezi jednotlivými aktivitami. Data lze uložit v souborovém systému, SQLite databázi, na webu, nebo do jakéhokoliv jiného trvalého úložiště, ke kterým může aplikace získat přístup. Prostřednictvím content provider, se mohou ostatní aplikace dotazovat nebo dokonce modifikovat data, pokud jim to content provider umožňuje.

Broadcast receivers

Broadcast receiver je komponenta, která reaguje na systémové broadcasty oznámením. Mnoho broadcastů přichází od systému např. obrazovka je vypnuta, baterie je vybitá, obrázek byl přijat apod. Všechny takovéto události mohou následně vyvolat patřičnou odezvu. A to jak ve formě výpisu na stavový řádek, spuštění aktivity, služby či jiné komponenty. Aplikace mohou využívat systémové broadcasty nebo vytvářet své vlastní.

2.1.3 Životní cyklus Aktivity

Životní cyklus Android aktivity zahrnuje soubor metod vestavěné ve třídě Activity, která poskytuje vývojáři rámec (Framework) pro řízení zdrojů. Tento rámec (Framework) dovoluje vývojářům, aby řídila stav každé aktivity v rámci aplikace. Životní cyklus aktivity, tak napomáhá vývojáři s poskytováním shodného rámce (frameworku), ve kterém, lze ovládat řízení zdrojů v aplikaci.



Obrázek č. 2 – Diagram životního cyklu aktivity, zdroj [5]

Stavy aktivity [5,6,7]:

onCreate()

Poté, co byla vytvořena instance aktivity je volána metoda onCreate. Tady si většina aplikací provádí inicializaci:

- Čtení v rozložení (layout),
- vytváření View instancí,
- vazba na data atd.

Jestli instance Activity není zničená (uvolněná), ani proces není zničen, není metoda onCreate() znovu volána. Je volána pouze v případě, kdy je vytvářena nová instance třídy Activity. Argument této metody je objekt Bundle, který obsahuje uložený stav aplikace. Není-li stav uložený, hodnota tohoto argumentu je null.

onStart()

Tato metoda je volána, když se aktivita stává viditelnou pro uživatele.

onResume()

Metoda je volána, když začne aktivita interakci s uživatelem, poté co byla v pozastaveném stavu. Aktivity mohou tuto metodu přepisovat, pokud potřebují k provádění úkolů po aktivitě začít přijímat vstup od uživatele.

onPause()

OnPause metoda je volána, když se instance jiné aktivity bude chtít stát viditelná a současná aktivita zastaví interakci s uživatelem.

onStop()

OnStop() metoda je volána, když aktivita již není zobrazena, protože další aktivita byla obnovena.

Je to protikladná metoda onStart(), Aktivita z pozastaveného stavu přechází na stav zastavená. Při níž je zrušeno všechno, co bylo vytvořeno v onStart(). OnStop() je protikladem metody onStart(), Activita z pozastaveného stavu přechází na stav zastavená. Vše co jsme v onStart() vytvořili, v ní zrušíme.

onRestart()

Tato metoda muže být volána po metodě onStop(), kdy je aktivita zastavena. OnRestart() je vždy následována metodou onStart(), kdy je aktivita znovu spuštěna. Aktivita by měla přepsat onRestart(), jestliže potřebuje provést nějaký úkol ještě předtím, než je volána onStart().

onDestroy()

Jedná se o poslední metodu, která je volána předtím, než je aktivita zničena. Poté, co je tato metoda volána, bude aktivita zničena a odstraněna z paměti. V této metodě je poslední možnost uložení stavu dat.

onSaveInstanceState()

Tato metoda poskytuje Android aktivity cyklu Framework k uložení dat, když dojde ke změně např. změna orientace obrazovky.

2.2 Bluetooth

Bluetooth je standart bezdrátové technologie, pro výměnu dat na krátké vzdálenosti, pomocí krátké vlnové délky rádiových přenosů v pásmu ISM 2400 až 2483MHz. Technologie vznikla ve firmě Ericssom v roce 1994. Bluetooth byl původně koncipován jako bezdrátová alternativa k datovým kabelům RS-232.

Umožňuje z pevných a mobilních zařízení vytvářet osobní sítě (PAN) s vysokou úrovní bezpečnosti. Bluetooth je řízen Bluetooth SIG, která má více než 17 000 členských společností v oblasti telekomunikace, výpočetní techniky, sítí a spotřební elektroniky. SIG dohlíží na vývoj specifikace, řídí program kvalifikace a chrání ochrannou známku. Aby mohlo být zařízení prodáváno jako Bluetooth , musí být způsobilé standardům definované SIG.

Bluetooth poskytuje bezpečný způsob připojení a výměny informací mezi zařízeními, jako jsou mobilní telefony, faxy, notebooky, osobní počítače, tiskárny, GPS přijímače, digitální fotoaparáty, herní konzole a jim podobná zařízení. [8,11]

2.2.1 Realizace

Bluetooth využívá rádiovou technologii FHSS, která rozdělí odesílaná data mezi 79 frekvencemi se vzájemným odstupem 1MHz v rozsahu 2400-2483,5MHz a během jedné sekundy je provedeno 1600 přeskoků. Rozsah se nachází globálně na nelicencovaném frekvenčním pásmu krátkého dosahu ISM 2,4Ghz.

Původně byla GFSK modulace jediným dostupným schématem, následně od uvedení Bluetooth 2.0+EDR může být používaná π/4-DQPSK a 8DPSK modulace mezi kompatibilními zařízeními. Pokud zařízení pracuje s GFSK říká se, že pracuje v BR modu, kde je možná okamžitá přenosová rychlost 1Mbit/s. Termín EDR se používá k popisu π/4-DPSK a 8DPSK schémat, jejichž přenosová rychlost je 2 a 3Mbit/s. Kombinace těchto dvou modů v Bluetooth technologii je klasifikována jako „BR/EDR rádio“.

2.2.2 Komunikace a spojení

Bluetooth podporuje jak dvoubodovou, tak mnohabodovou komunikaci. V piconet síti může jeden master komunikovat až se 7 zařízeními slave, ale ne všechny zařízení dosáhnou tohoto maxima. Zařízení si mohou měnit roli na základě dohody a slave se může stát master.

Jádro Bluetooth specifikace zajišťuje propojení dvou nebo více piconet sítí, které pak vytváří scatternet, v které některá zařízení hrají současně roli master v jedné piconet síti a v další jsou slave.

Všechny zařízení sdílejí hodiny od zařízení, které figuruje jako master. Výměna paketů je založena na základních hodinách, které jsou definované master zařízením a tikají v intervalech 312,5µs. Dvoje hodiny tvoří slot 625µs. V jednoduchém případě master odesílá v sudých slotech a v lichých přijímá, slave naopak v sudých přijímá a v lichých odesílá. Pakety mohou být dlouhé 1,3 nebo 5 slotů, ale za všech okolností začne master přenášet v sudých slotech a slave v lichých slotech. [10,11]

2.2.3 Typy výkonnostních tříd

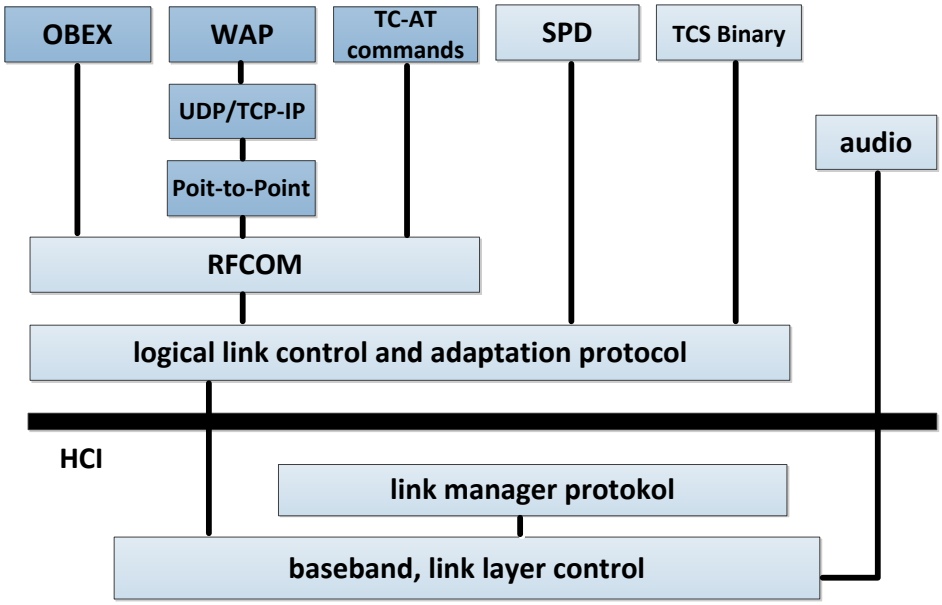
Bluetooth dělíme do tříd podle jejich výstupního výkonu. Jak je s tabulky č. 1 viditelné, dosah zařízení souvisí s jeho vysílacím výkonem. V mobilních telefonech se nejčastěji používá třída 2 s dosahem 10m.

Třída	Maximální povolený výkon		Přibližný dosah
	mW	dBm	
Třída 1	100	20	100m
Třída 2	2,5	4	10m
Třída 3	1	0	1m

Tabulka č. 1- Typy výkonnostních tříd u Bluetooth zdroj[12]

2.2.4 Architektura přenosových protokolů standardu Bluetooth

Na obrázku č. 3 je znázorněna architektura přenosových protokolů, jak ji definuje standard Bluetooth.



Obrázek č. 3 – Architektura přenosových protokolů zdroj [14]

Baseband, link layer control

Vrstvy baseband a link layer control umožňují fyzické spojení mezi zařízeními tvořící Bluetooth piconet. Oba okruhy je používají k přepojování paketů. Nabízejí dva typy fyzického propojení pomocí baseband paketů. Synchronní komunikace se spojováním (SCO) a asynchronní komunikace bez spojování (ACL). ACL pakety se používají pouze pro data, zatímco SCO pakety mohou obsahovat pouze zvuk nebo kombinaci zvuku a dat.

Link manager protocol

Vrstva link manager protocol (LMP) nastavuje spojení mezi jednotkami Bluetooth. Tato vrstva protokolu se zaměřuje na otázky bezpečnosti jako je autentizace, šifrování generováním, výměny a kontroly spojení a šifrovacích klíčů. Zabývá se rovněž kontrolou a vyjednáváním velikosti baseband paketu.

Host controller interface

Vrstva HCI poskytuje jak jednotné rozhraní, tak i metodu přístupu k hardwaru Bluetooth. Dále HCI obsahuje příkazové rozhraní k řadiči základního pásma, správy kanálu a přístup ke stavu hardwaru a kontrole registru.

Logical link control and adaptation protocol

Vrstva L2CAP poskytuje vyšším vrstvám služby pro spojové a nespojové datové přenosy. Podporuje vyšší úroveň multiplexu, který podporuje několik protokolů, jenž jsou definovány vyššími vrstvami Bluetooth protokolů. Především se jedná o protokoly RFCOMM,SDP a TSC Binary. Dále podporuje segmentaci a zpětné sestavování paketů, QoS komunikaci a skupiny.

Radio frequency communications port

Protokol RFCOMM tvoří základ pro nahrazení kabelové komunikace Bluetoothem. Jedná se o jednoduchý transportní protokol s dodatečnými opatřeními pro emulaci sériového portu RS-232 přes L2CAP část Bluetooth protokol stacku. Protokol poskytuje služby vyšším vrstvám, které používají pro přenos dat sériovou linku.

Service discovery protocol

Protokol SPD umožňuje klientské jednotce Bluetooth nalézat služby podporované dalšími zařízeními a jejich přidružené parametry. Součástí služeb je vyhledávání nově dostupných služeb v síti, ale také detekce služeb, jejichž poskytování bylo ukončeno. Veškeré služby jsou identifikovány jedinečným UUID. Krátkou 16bitovou formu UUID používají oficiální služby Bluetooth profilů. Jiné než oficiální používají 128bitové UUID

Telephony control – binary

Protokol TCS Binary je bitově orientovaný protokol. Definuje sestavení, řízení přenosové linky a přenos hlasu a dat mezi jednotkami Bluetooth.
[9,14]

2.2.5 Adaptované protokoly

Protokoly PPP, TCP / IP

PPP, TCP, UDP a IP jsou standardní internetové protokoly definované IETF. Jsou používané jako protokoly nižší vrstvy z WAP stacku.

OBEX

OBEX je volitelný protokol určený pro přenos dat a řídících informací. Je navržen k tomu, aby umožnil zařízením podporujícím infračervenou komunikaci, vyměnit širokou škálu dat. Protokol OBEX je nezávislý na transportním mechanismu a přenosovém programovém rozhraní. Využívá architekturu klient-server. Jako hlavní transportní protokol používá RFCOMM.

WAP/WAE

Bluetooth může být použit jako technologie nosiče pro transport mezi WAP klientem a blízkým WAP serverem. WAP pracuje na vrcholu Bluetooth stacku a používá PPP a sadu protokolů TCP/IP.

2.2.6 Bluetooth profily

Profily popisují, jak mohou být různé části specifikace použity ke splnění požadované funkce pro Bluetooth. Profil lze popsat jako vertikální řez protokolu stacku. Definuje možnosti každého protokolu, které jsou povinné pro profil. Definuje také rozsah parametru pro každý protokol.

Koncept profilu se používá ke snížení rizika problémů nekompatibility mezi zařízeními různých výrobců. Základní Bluetooth technologie je stejná, pouze se liší specifickým způsobem, jak je používána, definována a vyjasněna.

Základní Bluetooth profily:

GAP (Generic Accces Profile)

Poskytuje základ pro všechny ostatní profily. Vycházejí z něj další profily, tím je zaručena kompatibilita.

SDAP (Service Discovery Application Profile)

Profil SDAP popisuje, jak má aplikace využít SDP k zjištění služeb na vzdáleném zařízení.

SPP (Serial Port Profile)

Tento profil na Bluetooth zařízení, definuje emulaci RS-232 sériového rozhraní.

GOEP (Generic Object Exchange Profile)

Profil GOEP definuje role serveru a klienta a taktéž vytváří základ pro další profily zabývající se výměnou dat.

AVRCP (Audio/Video Remote Control Profile)

Tento profil je navržen k tomu, aby poskytoval standardní rozhraní k ovládání TV, hi-fi nebo jiného zařízení, aby jeden dálkový ovladač sloužil, k ovládání všech A/V zařízení, ke kterým má uživatel přístup.

GAVDP (Generic Audio/Video Distribution Profile)

GAVDP poskytuje základ pro A2DP a VDP, základ systémů určených k distribuci video a audio streamů pomocí technologie Bluetooth.

DUN (Dial-Up Networking)

Popisuje, jak má zařízení využívat modem pro připojení k telefonní síti. Je postaven na profilu sériového portu.
[13,14]

2.3. Wi-Fi

Wi-Fi je technologie, která umožňuje elektronickým zařízením bezdrátovou výměnu dat pomocí rádiových vln přes počítačovou síť, včetně internetového připojení. Wi-Fi Aliance definuje Wi-Fi jako "bezdrátový lokální síťový (WLAN) produkt, který je založený na normě IEEE 802.11". Pouze Wi-Fi produkty, které mají kompletní Wi-Fi Aliance interoperability certifikační zkoušky úspěšně mohou používat "Wi-Fi CERTIFIED" ochrannou známku.

Základem každé Wi-Fi sítě je jeden nebo více přístupových bodů AP. Přístupový bod AP umožňuje bezdrátovým stanicím připojení a komunikaci se zařízeními v různorodých sítích. Každé AP má své SSID, což je název pro WLAN, které může být veřejné, tak i skryté. Jeho maximální délka může být až 32 znaků.
Zařízení, které může používat Wi-Fi (jako je osobní počítač, herní konzole, smartphone , tablet, nebo digitální audio přehrávač) může být připojeno k síťovému prostředku, jako je internet přes přístupový bod bezdrátové sítě. Přístupový bod má rozsah cca 20 metrů v interiéru a většího rozsahu venku. [15,16,17]

Decentralizované typy bezdrátové sítě:

Ad-Hoc

Bezdrátová síť ad-hoc je decentralizovaný typ bezdrátové sítě, protože není závislá na již existující infrastruktuře, jako jsou například směrovače v kabelových sítích nebo přístupové body v bezdrátových sítích. U bezdrátových počítačových sítí je ad-hoc metoda, pro bezdrátová zařízení komunikující přímo mezi sebou.

K nastavení ad-hoc bezdrátové sítě, musí každý bezdrátový adaptér být nakonfigurován pro ad-hoc režim, oproti alternativnímu režimu infrastruktury. Kromě toho, musí všechny bezdrátové adaptéry v síti ad-hoc, používat stejný SSID a stejné číslo kanálu. Ad-hoc sítě obvykle mají malou skupinu zařízení v těsné blízkosti vedle sebe. Výkonnost se snižuje s přibývajícím počtem zařízení a velké ad-hoc sítě se obtížně spravují.

Sítě ad hoc dávají smysl, pokud je potřeba rychle vytvořit malou, bezdrátovou síť LAN s minimální investicí za vybavení. [18]

Wi-Fi Direct

Wi-Fi Direct, dříve známé jako Wi-Fi P2P, je standard, který umožňuje zařízením Wi-Fi připojení k sobě, bez nutnosti přístupového bodu. Wi-Fi Direct umožňuje zařízením přenášet data přímo mezi sebou, se značně sníženým nastavením. Technologie používá nejmodernější šifrování WPA2, které je dostatečně bezpečné. Výměna šifrovacích klíčů je díky funkci Wi-Fi Protected Setup (WPS) rychlá a jednoduchá. Zařízeni se musí před komunikaci spárovat, což je otázkou stisknutí tlačítka, podobně jako u NFC. Pokud již jsou zařízení spárována, spojí se pomocí speciálního rozpoznávacího paketu (Probe Request). Každé Wi-Fi Direct zařízení je schopné tento speciální paket odesílat i přijímat.

Wi-Fi Direct certifikované zařízení, lze použít pro všechny druhy aplikací jako je sdílení obsahu, synchronizace dat, hrání her, streamování audio a videa do televize, a další - v podstatě všechny věci, které lze dělat s Wi-Fi zařízeními dnes, jen snáz.
Největší výhodou technologie Wi-Fi Direct spočívá především ve vysoké přenosové rychlosti, dostatečně velkém dosahu a jednoduchosti použití. Předpokládá se, že v budoucnu nahradí Bluetooth především v oblasti přenosu dat.[19,20]

2.4. Standardy IEEE 802.11

2.4.1 Standart IEEE 802.11

V roce 1997, IEEE vytvořil první WLAN standard. Nazvali ho 802.11 podle názvu skupiny, která dohlíží na jeho vývoj. Umožňuje přenosovou rychlost až 2Mbit/s. Což je příliš pomalé pro většinu aplikací. Z tohoto důvodu se běžné 802.11 bezdrátové produkty již nevyrábí.

Používá tři možnosti řešení fyzické vrstvy:

- **DSSS**
- **FHSS**
- **Infračervený přenos**

Linková vrstva poskytovala následující služby:

- **Autentizace**
- **Asociace**
- **WEP**

2.4.2 IEEE 802.11b

IEEE rozšířila původní standard v červenci 1999, jejímž cílem bylo odstranit především nízkou přenosovou rychlost.

802.11b používá stejné frekvenční pásmo (2,4 GHz) jako původní standard 802.11, ale dosahuje vyšších přenosových rychlostí.

Díky novému způsobu kódování tzv. doplňkové klíčové kódování (Complementary Code Keying) na fyzické vrstvě, pracující v rámci DSSS je dosažena maximální přenosová rychlost až 11Mbit/s.

Přenosová rychlost se dynamicky mění, v závislosti na prostředí. Dochází ke snižování a zvyšování rychlostí v řadě 1, 2, 5,5 až 11Mbit/s. Maximální teoretická rychlost na fyzické vrstvě je 11Mbit/s.

2.4.3 IEEE 802.11a

Zatímco 802.11b byl ve vývoji, IEEE vytvořili druhé rozšíření k původnímu 802.11 standardu, nazvaný 802.11a. Vzhledem k tomu, že 802.11b získal popularitu mnohem rychleji, než tomu bylo u 802.11a, někteří lidé si proto myslí, že 802.11a byl vytvořen po 802.11b. Ve skutečnosti byly vytvořeny ve stejnou dobu. Vzhledem k vyšší ceně, se 802.11a obvykle nachází v obchodních sítích, zatímco 802.11b lépe slouží domácímu trhu.

802.11a dosahuje teoretickou přenosovou rychlost až 54Mbit/s a pracuje ve frekvenčním pásmu 5 GHz. Je zde použita metoda rozprostřeného spektra OFDM. Vyšší frekvence oproti 802.11b zkracuje rozsah 802.11a sítí. Vyšší frekvence také znamená, že signály 802.11a mají větší potíže pronikat zdmi či jinými překážkami.

Vzhledem k tomu, 802.11a a 802.11b využívají různé frekvence, obě technologie jsou vzájemně nekompatibilní.

2.4.4 IEEE 802.11g

V roce 2003 bych schválen standard 802.11g. Je navržen pro frekvenční pásmo 2,4 GHz stejně jako jeho předchůdce 802.11b. Maximální přenosová rychlost je stejná jako u standardu 802.11a a dosahuje rovněž rychlost 54Mbit/s. Využívá OFMD modulaci, ale také obsahuje modulaci DSSS, a to z důvodu zpětné kompatibility ze zařízeními pracujících na standardu 802.11b. Do sítě 802.11g se může připojit zařízení pracující na 802.11b standardu a rychlost se sníží ze zmiňovaných 54Mbit/s na 11Mbit/s.

2.4.5 IEEE 802.11n

Norma 802.11n byla schválena v září roku 2009. K dosažení vyšších přenosových rychlostí byla upravena fyzická vrstva a část linkové vrstvy. Rychlost dosahuje až 600Mbit/s namísto půdního maxima 54Mbit/s.

Používá technologie předchozích norem, které zároveň rozšiřuje především o technologii MIMO**,** která zlepšuje odolnost proti rušení. MIMO lze využívat bez ohledu na protokoly vyšších vrstev, jelikož pracuje na fyzické vrstvě. Propustnost sítě a rychlost lze jednoduše zvýšit přidáním antén. Standard 802.11umožňuje pracovat jak v 2,4 GHz, tak i v 5 GHz frekvenčním pásmu. K dosažení maximální propustnosti sítě je doporučováno použití 5 GHz frekvenčního pásma.
[21,22,23]

2.5 JSON

JSON (JavaScript Object Notation) je odlehčený formát pro výměnu dat, který je nezávislý na počítačové platformě. Tento formát je jednoduše čitelný i zapisovatelný člověkem, je také snadno strojově analyzovatelný i generovatelný. JSON je zcela nezávislý formát na jazyce, který je zároveň textový. Využívá konvence, které jsou dobře známé programátorům jazyků rodiny C jako např. C, C++, C#, Java, JavaScript, a další. Díky tomu je JSON ideálním jazykem pro výměnu.

Alternativou JSONu může být velmi podobný SDL, který rovněž vkládá své datové typy a pole do složených závorek, nebo do YAML, který aktivně používá odsazování a může obsahovat i komentáře na rozdíl od JSONu. K další alternativě JSONu patří bezesporu XML, který dokáže pojmout i kontext toho, co přenáší. XML průměrně obsahuje 40% samotných značek a jejich atributů na rozdíl od JSONu. JSON považujeme za jednodušší alternativu právě k zmiňovanému XML.

Základní typy JSONu jsou:

- Číslo.
- Řetězec.
- Boolean - true nebo false.
- Pole - seřazená sekvence hodnot oddělených čárkami a uzavřena v hranatých závorkách. Hodnoty nemusí být stejného typu.
- Objekt - neuspořádaná kolekce, klíč: hodnota. Párovací znak je „:“, který odděluje klíč a hodnotu. Objekty jsou odděleny čárkou a jsou uzavřeny ve složených závorkách.

Klíče musí být řetězce a měly by být odlišné.

- Null

Na vstupu může být libovolná datová struktura (číslo, řetězec, Boolean, objekt nebo z nich složené pole), ale na výstupu je vždy řetězec. [24,25,26]

3. Rešerše současného stavu a používaných modulů Wi-Fi a Bluetooth v zařízeních s OS Android

Zařízení, která umožňují, bezdrátový přenos dat pomocí Wi-Fi či Bluetooth je na trhu nespočet. Může se jednat o mobilní telefony, notebooky, tablety, osobní počítače, tiskárny, GPS přijímače, digitální fotoaparáty, herní konzole, televize a spousty dalších zařízení. Každý uživatel vlastnící nějaké z těchto zařízení může zjistit, zdali toto zařízení má integrované Wi-Fi nebo Bluetooth a jaký Wi-Fi standard či Bluetooth verzi podporuje. Zjistit výrobce modulu či čipu již, tak snadné není. Ve specifikacích zařízení se téměř neuvádí.

Při vyhledávání informací o současném stavu používaných modulů Wi-Fi a Bluetooth v mobilních telefonech a tabletech bylo zjištěno, že v těchto zařízeních se nenachází Bluetooth a Wi-Fi čip samostatně, ale obě tyto technologie se nachází v jednom čipu. Je to hlavně z důvodu úspory místa na základní desce přístroje, tak i z důvodu levnější výroby. Pouze starší nebo levné telefony, které nemají integrovanou Wi-Fi technologii, obsahují pouze Bluetooth čip.

V mobilních telefonech či tabletech, které byly vyrobeny cca před dvěma lety, se používaly především čipy, tedy jeden čip, který obsahuje jak Wi-Fi, tak i Bluetooth.
Výrobce zařízení si koupil tento čip od vybrané společnosti zabývající se návrhem případně i výrobou Wi-Fi a Bluetooth čipů. Výrobce musel nejprve zajistit návrh pro komunikaci s tímto čipem, a také musel na základní desce přístroje umístit i součástky, které zajišťují funkčnost nebytnou pro komunikaci.

V současnosti osazují výrobci své zařízení převážně moduly, ale není to pravidlem. Záleží od výrobce nebo od modelu telefonu, a také to, v jaké cenové relaci se nachází. Modul se skládá z malého plošného spoje, na kterém se nachází čip s dalšími potřebnými součástkami, které jako celek zajišťují funkčnost nezbytnou pro komunikaci.

Tím, že modul již obsahuje součástky, které zajišťují funkčnost čipu nezbytnou pro komunikaci, muže být prostor, který by tyto součástky zabíraly na základní desce použit k jiným účelům.

Mezi hlavní výrobce čipů Wi-Fi a Bluetooth pro mobilní zařízení patří bezesporu Broadcom, Qualcomm a Texas Instruments. K nejpoužívanějším výrobcům modulů pro mobilní telefony a tablety patří Murata a AzureWave.

Mobilní telefon	Modul \ Čip	Wi-Fi	Bluetooth	GPS	FM
Nexus One	- \ Broadcom BCM4329	802.11 a/b/g/n	2.1 + EDR	ne	ano
Sony Xperia Play	- \ Broadcom BCM4329	802.11 a/b/g/n	2.1 + EDR	ne	ano
Motorola XT910 Razr	- \Texas Instruments WL1285C	802.11 a/b/g/n	3.0	ano	ano
HTC One X	- \Qualcomm WCN3660	802.11a/b/g/n	4.0	ne	ano
Samsung Galaxy S3	Murata M2322007 \ Broadcom BCM4334	802.11 a/b/g/n	4.0	ne	ne
Nexus 4	Murata SS2908001 \ Broadcom BCM4330	802.11 a/b/g/n	4.0	ne	ne
HTC One	- \ Broadcom BCM4335	802.11a/b/g/n/ac	4.0	ne	ano

Tabulka č. 2 – Používané Wi-Fi a Bluetooth moduly v mobilních telefonech s OS Android

Pro zjištění jaký čip nebo modul byl použit v konkrétním zařízení, byly využity www stránky, které se zabývají opravu či demontáží různých zařízení zejména www.ifixit.com a www.chipworks.com.

Jak je z tabulky č. 2 patrné nejpoužívanějším výrobcem je firma Broadcom, kterou používá většina výrobců. Řešení od Texas Instruments používá především Motorola, ale ne u všech zařízení. Čipy od firmy Qualcomm používá malá část zařízení.

Mobilní telefony Samsung Galaxy S3, Nexus 4 i Iphone 5 používají moduly od firmy Murata, ale liší se použitými čipy.

Z tabulky č. 2 a 3 je rovněž patrné, že v čipech není obsaženo pouze Wi-Fi a Bluetooth, ale také FM rádio a v řešení od Texas Instruments dokonce i GPS. Firma Broadcomm řešení s GPS zatím nenabízí, ale vyrábí GPS čipy samostatně.

Mezi současnou špičku v čipech patří Broadcom BCM4335, který používá nové HTC One. Tento nový čip Broadcomu nabízí 802.11ac a současně kompatibilitu s 802.11a/b/g/n, má rovněž integrovaný Bluetooth 4.0 a FM radio.

Tablet	Modul \ Čip	Wi-Fi	Bluetooth	GPS	FM
Acer Iconia Tab (A500)	AzureWave AW-NH611 \ Broadcom BCM4329	802.11 b/g/n	2.1 + EDR	ne	ano
Samsung Galaxy Tab	Broadcom BCM4329	802.11 a/b/g/n	2.1 + EDR	ne	ano
Motorola XOOM	Broadcom BCM4329	802.11 a/b/g/n	2.1 + EDR	ne	ano
Samsung Galaxy Note	Broadcom BCM4330	802.11 a/b/g/n	4.0	ne	ano
Nexus 7	AzureWave AW-NH665 \ Broadcom BCM4330 (upraven)	802.11b/g/n	3.0	ne	ne

Tabulka č. 3 – Používané Wi-Fi a Bluetooth moduly v tabletech s OS Android

U tabletů je situace podobná, nepoužívanějším výrobcem je opět firma Broadcom. Pouze výrobci Acer a Asus (Nexus 7) používají moduly od firmy AzureWave, která používá čipy také od Broadcomu.

Všechny zařízení uvedené v tabulkách č. 1 a 2 používají OS Android. Nejstarší z uvedených zařízení v tabulce č. 1 a č. 2 je Nexus One, který byl na trh uveden lednu 2010 a naopak nejnovějším zařízením je HTC One, který byl na trh uveden v březnu 2013. [27,28,29,30,31,32,33,34]

4. Rozhraní pro práci s Bluetooth a Wi-Fi v OS Android

4.1 Rozhraní pro práci s Bluetooh v OS Android

Platforma Android obsahuje podporu Bluetooth stack, který umožňuje zařízení bezdrátově vyměňovat data s ostatními Bluetooth zařízeními. Aplikační framework poskytuje přístup k funkcí Bluetooth přes Android Bluetooth API. Tato rozhraní API umožní aplikacím bezdrátové připojení k jiným Bluetooth zařízením, umožňující dvoubodové a vícebodové bezdrátové funkce.

Android používá Bluetooth stack BlueZ, který původně vyvíjela firma Qualcomm. Tento stack používá rovněž **OS** Linux.

Pomocí Bluetooth API, může Android aplikace provádět následující:

- Vyhledat další zařízení Bluetooth.
- Dotázat se lokálního Bluetooth adaptéru na spárovaná zařízení Bluetooth.
- Stanovení RFCOMM kanálů.
- Připojení k jiným zařízením přes objevené služby.
- Přenos dat do a z jiných zařízení.
- Správa vícenásobného připojení.

Všechny Bluetooth API jsou k dispozici v balíčku android.bluetooth. Zde je přehled některých tříd a rozhraní, které byly implementovány:

BluetoothAdapter

Představuje místní adaptér Bluetooth. V BluetoothAdapteru je vstupní bod pro všechny Bluetooth interakce. S pomocí tohoto nástroje lze zjistit další zařízení Bluetooth, dotázat se na seznam spárových zařízení, vytvořit instanci BluetoothDevice se známou MAC adresou a

vytvořit BluetoothServerSocket k naslouchání komunikace z jiných zařízení.

BluetoothDevice

Představuje vzdálené Bluetooth zařízení. Používá se k požadavku o spojení se vzdáleným zařízením prostřednictvím BluetoothSocket nebo k dotazu informace o zařízení, jako je jeho název, adresa, třída a stav.

BluetoothSocket

Představuje rozhraní pro Bluetooth soket. Je to bod spojení, který umožňuje aplikaci vyměňovat data s jiným Bluetooth zařízením přes InputStream a OutputStream.

BluetoothServerSocket

Představuje otevřený server soket, který naslouchá příchozím požadavkům. Aby se mohly spojit dvě Android zařízení, musí jedno z nich otevřít server socket s touto třídou. Když vzdálené Bluetooth zařízení předává požadavek na připojení, BluetoothServerSocket vrátí připojený BluetoothSocket, když je přijato. [5]

4.2 Rozhraní pro práci s Wi-Fi v OS Android

Android jako takový obsahuje 2 hlavní API pro práci s Wi-Fi:

android.net.wifi

Poskytuje třídy pro správu funkce Wi-Fi zařízení. Wi-Fi API poskytuje prostředky, kterými mohou aplikace komunikovat s nižší úrovní bezdrátového stacku, který poskytuje přístup k síti Wi-Fi. Téměř všechny informace ze zařízení jsou žadateli k dispozici, včetně rychlosti linky připojené sítě, IP adresy, stav vyjednávání a další. Navíc obsahuje informace o dalších sítích, které jsou k dispozici. Mezi jiné API funkce patří možnost skenování, přidávání, ukládání, ukončení a inicializování připojení Wi-Fi.

Zde je přehled některých tříd:

- ScanResult - Popisuje informace o detekovaném AP.
- WifiConfiguration -Třída zastupující nakonfigurovanou Wi-Fi sít, včetně
- konfigurace zabezpečení.
- WifiConfiguration.Status - Možný stav konfigurace sítě.
- WifiInfo - Popisuje stav každého Wifi připojení, které je aktivní nebo je v procesu sestavování.

- WifiManager - Tato třída poskytuje primární API pro správu všech aspektů Wi-Fi připojení.

android.net.wifi.p2p

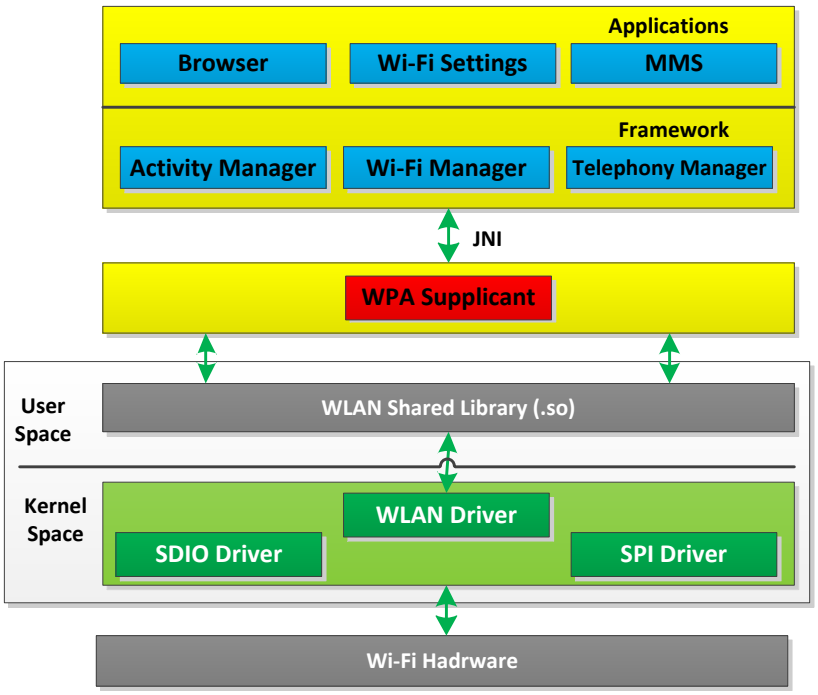
Obsahuje třídy pro vytváření peer-to-peer (P2P) připojení s Wi-Fi Direct. Pomocí těchto rozhraní, můžeme zjistit a připojit se k jiným zařízením. Pokud i tato zařízení podporují Wi-Fi Direct, pak mohou tato zařízení komunikovat přes toto rychlé připojení a na mnohem větší vzdálenosti než kdyby komunikovali přes Bluetooth.

Primární třída, se kterou je potřeba pracovat, je WifiP2pManager, kterou lze získat voláním getSystemService(WIFI_P2P_SERVICE).

WifiP2pManager obsahuje API, které vám umožní:

- Inicializace aplikace pro p2p spojení voláním initialize ().
- Zjištění okolních zařízení voláním discoverPeers ().
- Začne P2P připojení voláním connect ().
- A další.

Pro komponentu Wi-Fi jsem použil pouze první API z balíku android.net.wifi, jelikož Wi-Fi Direct je především určen ke komunikaci dvou zařízení, proto nebylo API z balíku android.net.wifi.p2p použito. [5]



Obrázek č. 4 – Architektura Wi-Fi v OS Android [35]

Architektura Wi-Fi v OS Android

Popis vrstev [35]:

- **Wi-Fi Settings** - Aplikace, prostřednictvím které může uživatel povolit/zakázat Wi-Fi nebo se připojit/odpojit k sítí, popřípadě spravovat Wi-Fi sítě.
- **Wi-Fi Manager** - Jedná se o službu a framework komponent. Slouží aplikacím k dotazování Wi-Fi funkcí vztahující se k ovladači. Také spravuje připojení k síti a spravuje Wi-Fi sítě.
- **Wifi Service** -Je také součástí frameworku, není zobrazen v diagramu. Ve skutečnosti řídí Wi-Fi příkazy od různých klientů (aplikací) a vykonává Wi-Fi příkazy k ovladači pomocí rozhraní JNI.
- **WPA Supplicant** - Je určen k provádění bezpečnostních metod jako je WPA,WPA2 a WEP a pro klíčové vyjednávání s Wi-Fi ovladačem. Je navržen jako obyčejná komponenta pro různé platformy, jako je Windows, Linux a iOS. Daemon hostapd je spuštěn v každém systému založeným na Linuxu, který využívá WPA Supplicant pro bezpečnostní metodu implementace a je odpovědný za spojení a odpojení s AP.

- **WLAN Shared Library** - Je k dispozici od dodavatele (třetí strana, která poskytuje Wi-Fi řešení) a bude mít rozhraní pro interakci s Wi-Fi ovladačem a rozhraním API. Bude načtena v uživatelském prostoru.
- **SDIO** - Slouží pro interakci s Wi-Fi čipem a pro přenos dat mezi čipem Wi-Fi a OS.
- **Wi-Fi hardware** - Čip Wi-Fi spolu s ovladači jádra.

5. Multiplayer řešení pro OS Android

Na trhu se nachází celá řada herních enginů pro OS Android, ale již málo z nich má integrovanou komponentu nebo engine pro vytvoření multiplayer her. Herní enginy jsou především určeny k tvorbě her. Pokud vývojáři chtějí, aby jejich hra umožňovala multiplayer, většinou sáhnou po hotovém řešení, které již je na trhu, než aby vytvářeli své vlastní řešení. Významem této kapitoly je seznámení s případnými komponentami nebo spíše se síťovými enginy s podporou multiplayer her pro OS Android, které se nachází na trhu.

AndEngine

Android 2D herní engine OpenGL AndEngine je open source Android herní engine, který je zdarma a je určen pro platformu Android a je na tuto platformu optimalizován. Je kompatibilní s OS Android od verze 1.6. Tento engine umožňuje rozdělení obrazovky, vytváření živých tapet na plochu a také podporuje multitouch. Využívá open source dvourozměrný fyzikální engine Box2D, který je napsán C++. Pro tento engine existuje i multiplayer rozšíření, ale pouze přes Wi-Fi síť. Na trhu je mnoho herních enginů, které jsou poskytovaný zdarma, ale AndEngine patří v této skupině mezi nejpoužívanější. [36]

Mages

Mages je multiplayer klient/server herní engine pro Android. Umožňuje vývojářům vytvářet internetové multiplayer hry pro více hráčů implementací pouze základní herní logiky a GUI pomocí výkonného API. Vývojáři mohou znovu použít efektivní Comet-based engine síťový protokol pro běžné herní úkoly, jako je přihlášení k hernímu serveru, vyhledání aktivního seznamu hráčů, seznam dostupných založených her, vytvoření nové hry, připojení již existující hře, chatování s oponenty apod. Comet je webový aplikační model, který nabízí interakci v reálném čase, pomocí trvalého nebo dlouhodobého HTTP spojení mezi serverem a klientem. Jak již bylo zmíněno, jedná se o multiplayer herní engine určený pro internetové hry, s minimální latencí a minimálními nároky na data si vystačí i GPRS připojením. Je poskytován zdarma.[37]

RakNet

RakNet je multiplatformní C + + a C # herní síťový engine. Je navržen tak, aby podával vysoký výkon, šel snadno integrovat a poskytoval kompletní řešení pro hry a další aplikace. RakNet podporuje všechny nejpopulárnější herní platformy a je integrován do mnoha middleware řešení. RakNet podporuje Windows, PlayStation 3, Xbox 360, PlayStation Vita, Linux, Mac, iOS, Android, Google Native Client, Windows Phone 8 a Windows 8 Store.

RakNet spolupracuje s některými z největších jmen v herním průmyslu např. Sony Online Entertainment, Unity, Minecraft, Maxis, Gamebryo a dalšími. Cena licence muže být zdarma nebo může dosáhnout až sumy 50000$. Záleží na podporované platformě, podpoře, ale i na příjmech ze hry. [38]

Photon

Photon je komunikační vrstva bez nadbytečné režie. Photon garantuje nejnižší časové odezvy v mobilních sítích a to z něj dělá ideální síťový engine pro iOS a Android, stejně jako pro Windows Phone. To je důvod, proč je největším světovým sociálním ekosystém pro mobilní hry. OpenFeint s více než 110 milionů registrovaných uživatelů si zvolil Photon jako svůj síťový engine. Jádro Photonu je postaveno z výkonnostních důvodů na nativním C / C + +, zatímco vlastní serverový kód je napsán v C # /. NET.

Photon je k dispozici na všechny populární platformy jako iOS, Android, Adobe Flash, HTML5, BlackBerry, DotNet, Unity, Marmalade. Cena licence se mění v závislosti na současném počtu připojených zařízení, a také je-li licence vázána k určité aplikaci. Cena se pak může vyšplhat až 1500$ za měsíc. [39]

Skiller

Skiller vyvíjí a poskytuje mobilní sociální herní platformu, která umožní vývojářům snadno nasadit komplexní sociální a multiplayer hry. Skiller umožňuje rychlý a jednoduchý způsob, jak vytvořit multiplatformní real-time a turn-based hry pro mobilní telefony. Podporuje Android, Windows Phone 7, Java ME, Unity a další. Zjednodušuje proces vytváření her pro více hráčů a umožňuje rychlé vytvoření sofistikovaných turn-based a real-time her.

Řešení od Skiller umožňuje hrát hry nezávislé na geografické poloze hráče a nezávislé na platformě. Toto řešení klade důraz na jednoduchost použití jako svou nejvyšší prioritu, takže je snadné a spolehlivé pro vývojáře, tak i pro hráče. Je poskytován zdarma, ale pokud hra umožňuje nakup virtuálního zboží nebo peněz, tak vývojář dostane ze zisku 70%. [40]

6. Analýza a návrh

6.1 Analýza

Při samotné analýze se bylo třeba zaměřit, k jakým úkonům budou tyto komponenty primárně používány a kdo s nimi bude pracovat.

Jelikož se má jednat o samostatné komponenty, které budou moci být použity v různých aplikacích, ke spojení a komunikaci více zařízení v lokální síti pomocí technologií Wi-Fi a Bluetooth, bylo jasné, že návrh bude rozdělen do více částí. Tyto komponenty mají přinést vývojářům rozšíření pro jejich aplikace, které jim zajistí spojení a komunikaci více zařízení v lokální síti pomocí Wi-Fi a Bluetooth.

Dále bylo potřeba nastudovat obě technologie, jak Wi-Fi, tak i Bluetooth a především zjistit možnosti API v OS Android pro práci s nimi.

Obě komponenty by měly umožňovat komunikaci dvou a více zařízení, z čehož je patrné, že alespoň jedno z těchto zařízení bude sloužit jako server, ke kterému se budou připojovat klienti, kteří budou se serverem komunikovat pomocí jednotného protokolu.

Bylo se třeba také zamyslet, v jakých oblastech se budou tyto komponenty lišit a v čem se budou naopak shodovat. Hlavním důvodem je následná úspora času při implementaci.

6.2 Návrh

Návrh komponent je rozdělen do 5 částí. Nechybí ani porovnání v jakých oblastech budou komponenty shodné a v jakých se budou naopak lišit, aby byla zajištěna znovu použitelnost kódu. Čím méně se budou obě komponenty od sebe lišit, tím bude mít vývojář využívající obě komponenty usnadněnou práci a orientaci v kódu.

Rozdělení návrhu (shoda[=]/odlišnost[≠] v návrhu komponent):

1. Klient-server (navázání, komunikace a ukončení spojení).
   Wi-Fi ≠ Bluetooth
2. Aktivity a vlákna.
   Wi-Fi = Bluetooth
3. Upozornění aktivity na změnu.
   Wi-Fi = Bluetooth
4. Návrh základního uživatelského rozhraní.
   Wi-Fi ≠ Bluetooth

5. Komunikační protokol.
   Wi-Fi = Bluetooth

6.2.1 Návrh klient-server (navázání, komunikace a ukončení spojení)

Návrh klienta a serveru pro Wi-Fi komponentu

Na začátku bylo uvažováno o propojení zařízení pomocí Wi-Fi ad-hoc sítě, která by byla decentralizována, umožňovala by spojení Wi-Fi zařízení bez AP. Po nastudování Android API pro práci s Wi-Fi bylo zjištěno, že Android API vytváření Wi-Fi ad-hoc sítí neumožňuje. Pokud je ovšem na zařízení s OS Android proveden root, který umožňuje spuštění aplikací jako superuživatel, provádění úprav na systémovém oddílu a dalších operací, je pak díky úpravě konfiguračních souborů umožněno založení Wi-Fi ad-hoc sítě. Jelikož má být komponenta určena pro aplikace nebo hry, které pak budou umístěny na online distribuční službu Google Play, varianta s rootlým Android zařízením a změnou konfiguračních souborů nebyla vhodným řešením. Především z důvodu omezení používaní výsledné aplikace nebo hry pouze na rootlé Android zařízení, protože ne každý uživatel zařízení s OS Android má provedenou tuto úpravu, kvůli které by vývojář přišel o velkou část svých případných zákazníků.

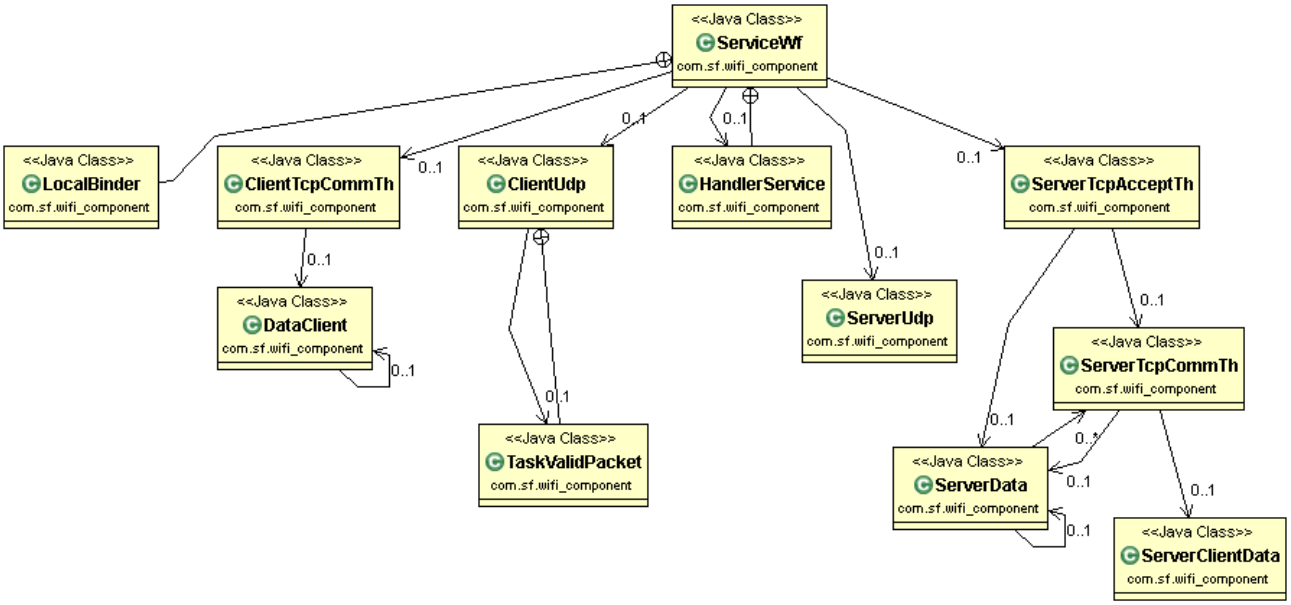
Od Androidu 4.0 (API 14) je k dispozici Wi-Fi Direct, který umožňuje propojení Wi-Fi zařízení bez AP. Tato technologie umožňuje přímé propojení pouze dvou zařízení. Samsung ve svých zařízeních sice nabízí propojení více zařízení bez AP pomocí Wi-Fi, ale jedná se o jeho softwarové řešení určené exkluzivně pro jeho zařízení.

Z tohoto důvodu bylo zvoleno propojení zařízení přes AP, které umožní připojení většího množství zařízení. Naopak jeho nevýhodou bude závislost na AP, tudíž nebude možné propojení zařízení v místech, kde se AP nenachází nebo nemá dostupnost. Tento nedostatek řeší Bluetooth komponenta.

Společné požadavky, které jsou kladeny na klienta a server jsou bezesporu zapnutí Wi-Fi, připojení k AP, spojení a komunikace. Při návrhu se bylo potřeba zamyslet, jak se klient dozví o již spuštěném serveru, ke kterému by se mohl připojit.

Nejsnazším řešením je přímé zadání IP adresy serveru, ale tento způsob není moc uživatelsky přívětivý. Uživatel, jehož zařízení slouží jako server, by musel předat svou IP adresu ostatním uživatelům, kteří se chtějí připojit jako klienti a ti by pak museli zadávat IP adresu serveru ručně.

Z tohoto důvodu bylo pro odesílání a příjem informací o serveru zvažováno nad použitím broadcast nebo multicast kanálu. Tyto kanály používají k šíření paketu protokol UDP, který není spolehlivý, což znamená, že může dojít ke ztrátě nebo doručení zprávy ve špatném pořadí.



Obrázek č. 5 - UML diagram Wi-Fi komponenty

Byl vybrán multicast kanál, protože v některých sítích může být šíření broadcast kanálem zakázáno. Navíc multicast umožňuje komunikaci s konkrétní skupinou posluchačů, kteří poslouchají na předem určené IP multicast adrese. Broadcast je šířen všem posluchačům v dané podsíti.

Pro komunikaci mezi klientem a serverem byl použit TCP protokol, který na rozdíl od UDP protokolu garantuje spolehlivé doručování paketů ve správném pořadí. V API OS Androidu je TCP komunikace reprezentována pomocí Sockets.

- **Požadavky na server Wi-Fi -** Požadavků kladených na server je mnoho. Především však by měl umožňovat připojení a komunikaci s více klienty najednou. Ale také posílání zpráv všem klientům nebo jednomu konkrétnímu klientovi. Dále pak ukončení spojení s odpojeným klientem nebo při ukončení serveru odpojení všech klientů. Aby byl klienty nalezen, měl by odesílat informace o sobě samém, v mém případě pomocí multicastu.

- **Požadavky na klienta Wi-Fi -** Klient by měl umožňovat vyhledávání, připojení a komunikaci se serverem. Následně by měl umět bezpečné odpojení od serveru, a to jak ze strany klienta, tak ze strany serveru.

Návrh klienta a serveru pro Bluetooth komponentu

Mezi Wi-Fi a Bluetooth komponentou by nebyly téměř žádné rozdíly, kdyby používaly stejné API pro práci s těmito čipy nebo moduly. Protože jsou obě tyto technologie odlišné a mají především i odlišné API, jsou tyto komponenty odlišné právě ve spojení a komunikaci.

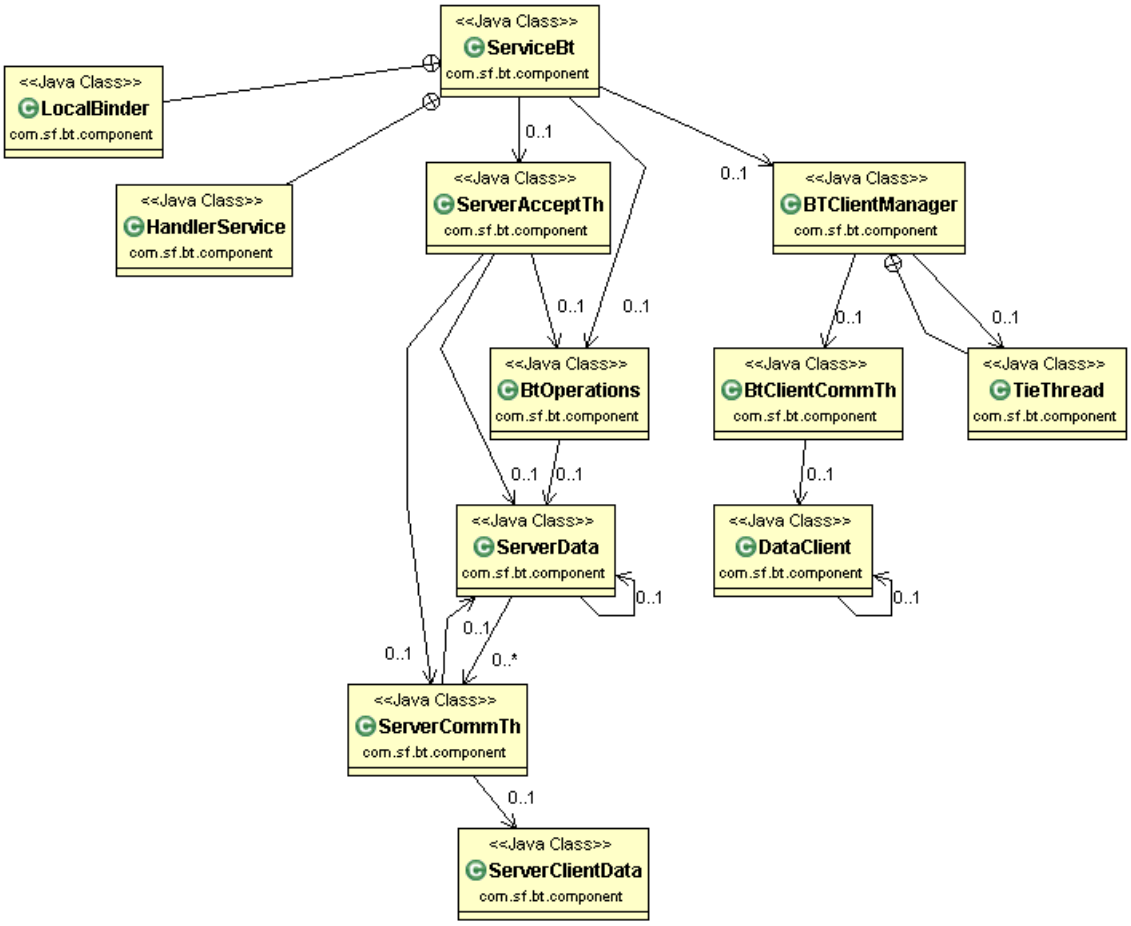
Při návrhu Bluetooth klienta a serveru pro Bluetooth komponentu se bylo třeba zamyslet nad možnostmi API v OS Android pro práci s Bluetooth, ale také nad technologickými možnostmi samotné technologie Bluetooth.

Technologie Bluetooth má omezení především v počtu zařízení, které se mohou připojit k master zařízení, které v tomto případě figuruje jako server. K jednomu master zařízení se může připojit maximálně sedm slave zařízení. Jedná se o omezení ze strany technologie, se kterým bylo počítáno již při samotném návrhu.

Při první komunikaci s dalším zařízením Bluetooth, je potřeba provést proces nazývaný jako párování. Tento proces umožní kontrolu nad zařízením, které se může připojit k danému Bluetooth a navázat komunikaci bez zásahu uživatele, pokud je zařízení v dosahu. Od verze OS Android 2.3.3 (API10), lze provádět spojení přes nezabezpečený RFCOMM, který umožňuje Bluetooth zařízení komunikaci bez nutnosti párování. Tento způsob komunikace přes nezabezpečený RFCOMM je vhodný především pro dočasné spojení jako např. pro přenos souborů, fotografií a dalších. Není vhodný pro dlouhodobé propojení. Z toho důvodu komunikace probíhala přes zabezpečený RFCOMM, který sice vyžaduje párování, ale za to bude komponenta použitelná i pro nižší verze OS Android.

Každá služba v Bluetooth je reprezentována jedinečným UUID. Bluetooth technologie používá SDP k zjištění služeb, které jsou zařízením podporovány a jaké parametry se mají použít k připojení k těmto službám. Aby se mohly dvě zařízení spojit, je nutné, aby byly spárované, a obě musí mít stejné UUID. Má-li ovšem server komunikovat z více než jedním klientem, musí server naslouchat více UUID. Každé spojení mezi serverem a klientem je v Bluetooth bráno jako služba, která je reprezentována jedinečným UUID. V rámci jedné služby tudíž nemohou komunikovat víc než dvě zařízení. Proto, pokud server komunikuje s více než jedním klientem, musí mít k dispozici potřebný počet jedinečných UUID, aby mohl tyto klienty obsloužit. Rovněž klienti musí mít k dispozici totožná UUID jako server. Pokud se bude chtít klient připojit k serveru, musí znát nejen jeho MAC adresu, ale musí rovněž používat stejné UUID, na kterém server naslouchá.

Komunikace mezi klientem a serverem bude probíhat pomocí protokolu RFCOMM, který v Bluetooth poskytuje emulování sériového portu RS-232. V API OS Android Bluetooth je komunikace reprezentována pomocí Bluetooth Sockets.



Obrázek č. 6 - UML diagram Bluetooth komponenty

- **Požadavky na server Bluetooth -** Požadavky kladené na server komponenty Bluetooth jsou téměř totožné s požadavky, jaké jsou kladeny na server komponenty Wi-Fi. Rozdíl je v tom, že je znám maximální možný počet klientů, kteří se mohou k serveru připojit. Dalším rozdílem je šíření informací o serveru, protože Bluetooth neumožňuje multicast ani broadcast.

- **Požadavky na klienta Bluetooth -** Taktéž požadavky kladené na klienta komponenty Bluetooth jsou téměř totožné s požadavky, jaké jsou kladeny na klienta komponenty Wi-Fi.

6.2.2 Návrh Aktivity a vlákna

Pokud pracujeme v rámci jedné aktivity s vláknem, které bylo v této aktivitě spuštěno, není problém vlákno také v této aktivitě zastavit. Pokud bychom však potřebovali pracovat s tímto vláknem ve více aktivitách, nastává problém s předáváním vlákna do další aktivity. Cílem bylo umožnit serveru přijímání nových klientů a rovněž komunikaci s klienty, či naopak umožnit klientovi připojení k serveru. Mimo to cílem bylo taktéž umožnit komunikaci se serverem nevázanou pouze k jedné aktivitě.

Řešení bylo nalezeno v Service(služba), což je aplikační komponenta v OS Android, která může provádět dlouhotrvající operace na pozadí a neposkytuje uživatelské rozhraní.

Kromě toho, že Service v OS Android umožňují spuštění operací na pozadí na dobu neurčitou, mohou také poskytovat klient-server rozhraní, přes které může klient komunikovat. Klientem se myslí aktivita a serverem Service.

Services, které jsou serverem v klient-server rozhraní, jsou označovány jako Bound Services. Bound Services mohou být vytvořeny buď lokálně v daném procesu, nebo odlehlém procesu. Ty, které jsou vytvořeny lokálně, slouží v rámci jedné aplikace. Servises vytvořené v daném procesu díky AIDL rozhraní a meziprocesorové komunikaci (IPC) mohou sloužit více aplikacím.

Pro potřeby obou komponent byla použita Bound Service bez AIDL rozhraní, jelikož není potřebná komunikace mezi více aplikacemi.

6.2.3 Návrh upozornění aktivity na změnu

Další částí, kterou bylo potřebné při návrhu vyřešit, bylo:
Jak aktivity upozornit z komunikačního vlákna na příchozí zprávy a reagovat, tak na změny, jež jsou očekávány, po přijmutí těchto zpráv?

V tomto případě bylo uvažéváno nad použitím BroadcastReceiveru, který je součástí API OS Android nebo nad externí knihovnou EventBus určenou pro OS Android.

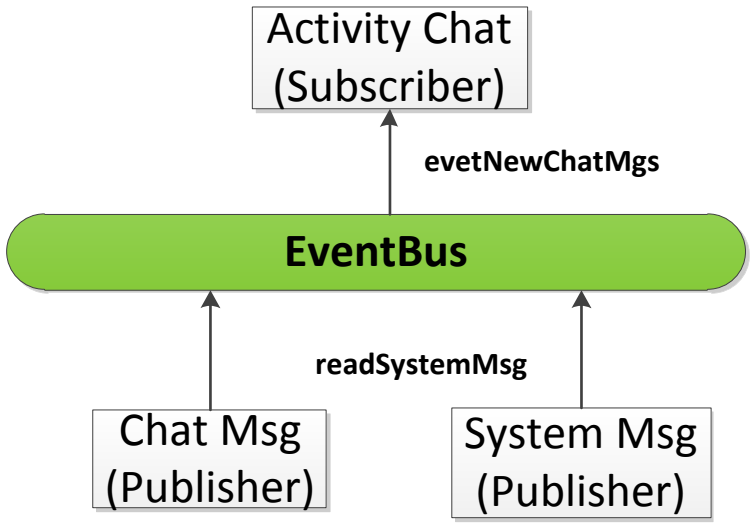
BroadcastReceiver je komponenta v OS Android, která umožňuje přihlásit k odebíraní různých receivers (přijímačů), na které může podle potřeby zareagovat. Tyto receivers mohou být jak systémové, tak si lze vytvořit své vlastní. Každý receiver, na který má BroadcastReceiver reagovat je nutné v metodě onResume() zaregistrovat. Při zničení aktivity se musí receiver zase v metodě onPause() odregistrovat. Posílání Intents probíhá díky metodě Context.sendBroadcast(), do které se jako parametr vloží předem vytvořený Intent s žádanou akcí.

Knihovna EventBus je optimalizována pro OS Android. Umožňuje publikování a odběr události ze sběrnice. Typickým příkladem použití pro Android aplikace je „lepení“ aktivit, fragmentů a vláken běžících na pozadí dohromady. Konvenční zapojení těchto prvků se často

složitě zavádí a je náchylné k chybám. EventBus odděluje událost vysílání i přijímání, a tím zjednodušuje komunikaci mezi aplikačními komponenty. Celkově obsahuje méně kódu a nemusí se implementovat jednotné rozhraní. Účastníci v EventBus, uplatňují metody zpracování událostí a registrování ke sběrnici. Při použití musí EventBus registrovat a odregistrovat své posluchače.

Producent posílá své události na sběrnici a posluchači pokud jsou zaregistrování k této sběrnici pak mohou tyto události přijímat.

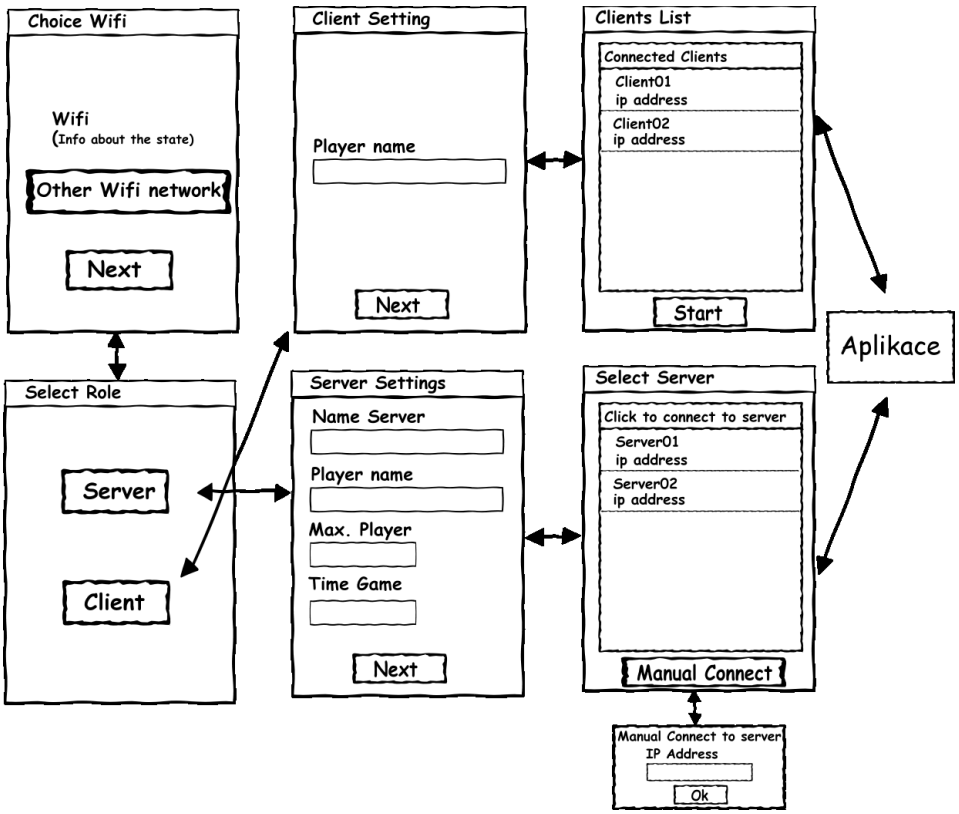
I přes výhody, které knihovna EventBus nabízí, bylo zvoleno řešení, které nabízí BroadcastReceiver. Především z důvodu, že je součástí API OS Android.



Obrázek č. 7 - EventBus

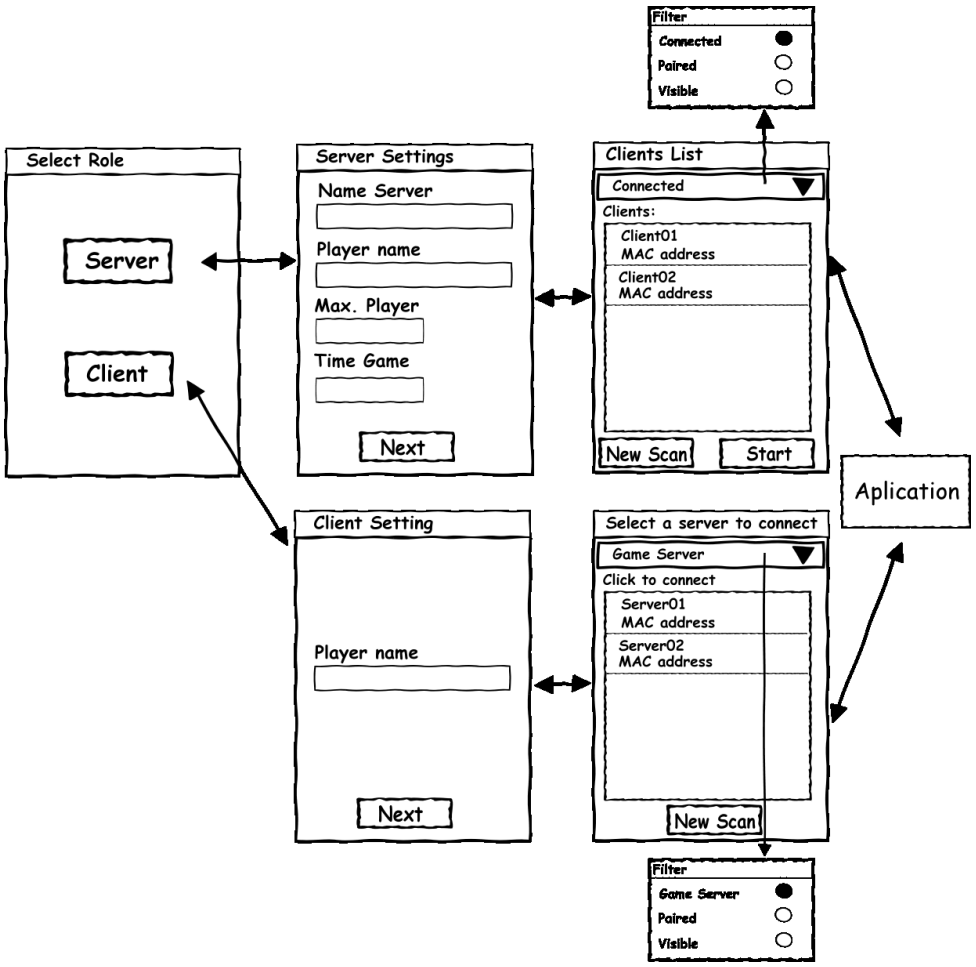
6.2.4 Návrh základního uživatelského rozhraní

Při návrhu uživatelského rozhraní (dále jen „UI“), bylo potřebné brát v úvahu, komu bude komponenta poskytována. Tyto komponenty budou poskytovány vývojářům, kteří je budou moci používat ve svých aplikacích, které budou zpřístupněny pro uživatele. Z toho vyplývá, že UI není určeno přímo pro uživatele, ale pro vývojáře, kterým má sloužit jako základ, z kterého mohou vycházet.



Obrázek č. 8 – Návrh UI komponenty Wi-Fi

Proto je u návrhu UI kladen důraz na jednoduchost a přehlednost, ale také, aby UI využilo všech funkcí, které mu Wi-Fi, čí Bluetooth komponenta nabízí. K návrhu a tvorbě UI aplikace pro OS Android je možné použit dvou způsobů, které se navzájem nevylučují. První je definice v XML a druhý je tvorba UI z aplikace. V tomto případě byla zvolena první možnost a tou je definice z XML. U komponent Wi-Fi a Bluetooth se návrh UI liší pouze v několika obrazovkách, jinak jsou shodné.



Obrázek č. 9 – Návrh UI kompomenty Bluetooth

6.2.5 Návrh komunikačního protokolu

K tomu, aby klienti mohli komunikovat se serverem a naopak, nestačí pouze zajištění spojení mezi nimi. Je nutné, aby měli stejný protokol, který budou používat ke komunikaci mezi sebou. Taktéž je nutné, aby měli stejný formát pro výměnu dat. Otázkou bylo, jaký formát pro výměnu dat zvolit. Nabízely se dva formáty, a to XML a JSON-em.

Oba tyto formáty se používájí k výměně dat mezí aplikacemi, především však pro výměnu informací na webu. JSON a XML mají mnoho podobných vlastností, ale XML nabízí více možnosti než JSON, ale za to má složitější strukturu. Ve výsledku více datově náročné.

Formát pro výměnu dat použitý pro tyto komponety měl splňovat tyto kritéria: jednoduchost, čitelnost a nízkou datovou náročnost. Rozhodnutí padlo na JSON, to a předším díky jeho datové úspoře, která může být až 40% nižší než u XML. JSON totiž neobsahuje párové značky a atributy.

Komunikační protokol navrhovaný pro obě komponenty byl rozdělen do tří zakladních typů, které mohou být dle potřeby vyvojáře upravovány nebo mohou být vytvářeny zcela nové typy zpráv.

Typy zpráv:

Systémové

V těchto zprávách jsou přenášeny data týkající se stavu a informací o serveru a klientech. Konkrétně, zda-li je server plný či nikoli, je-li klient připraven, informace o spuštění a zastavení serveru, a také dotazání na jmená klientů a serveru.

Chatové

Tyto zprávy, jak již název napovídá slouží k přenosu chatových zpráv, které obsahují jméno odesilatele a samotný text chatové zprávy.

Aplikační

Aplikační typy zpráv nebyly specifikovány. Tyto zprávy bude definovat samotný vývojář, který bude komponenty používat, a které budou sloužit pro přenos aplikačních dat, potřebných k funkčnosti celé aplikace.

7. Implementace

7.1 Implementace klienta a serveru pro Wi-Fi komponentu

Během implementace bylo postupováno, podle předem navržených UML diagramů. Během realizace byly ještě mírně upraveny a došlo k jejich rozšíření (finální UML diagram se nachází na obrázku č. 5). Aby mohlo vůbec dojít ke spojení více zařízení, nejdříve musí být Wi-Fi zařízení zapnuto a připojeno k AP. K tomu potřebuje aplikace v OS Android potřebná oprávnění, které musí být uvedeny v souboru AndroidManifest.xml. Tento soubor obsahuje každá Android aplikace a nachází se v kořenovém adresáři aplikace. Je generován automaticky a poskytuje všechny informace vztahující se k úspěšnému provedení všech funkcí přítomných v aplikaci.

K umožnění potřebné funkcionality Wi-Fi ,byly přidány do Androidmanifest.xml následující oprávnění:

- *android.permission.ACCESS_WIFI_STATE,*
- *android.permission.CHANGE_WIFI_STATE,*
- *android.permission.ACCESS_NETWORK_STATE.*

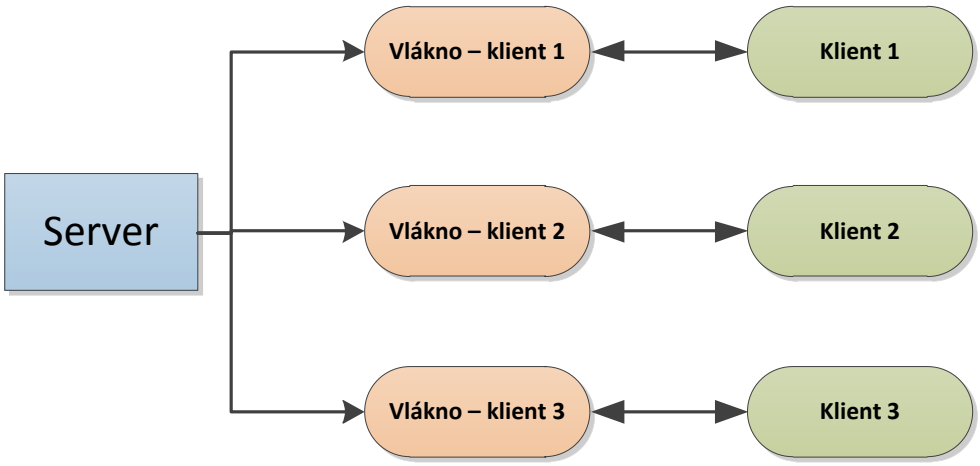
K zapnutí a správě Wi-Fi se v OS Android používá třída WifiManager. Do objektu třídy Wifimanager se vloží instance na službu WIFI_SERVICE a pomocí metody setWifiEnabled(), která má parametr boolean hodnotu, dojde k zapnutí nebo vypnutí Wi-Fi.

Poté, co byla zapnutá Wi-Fi, scházela možnost výběru AP k jakému se má Wi-Fi připojit, pokud je k dispozici více Wi-Fi sítí. K tomuto účelu byla použita systémová aktivita *ACTION_WIFI_SETTINGS,* která zobrazí nastavení, které umožňuje konfiguraci Wi-Fi.

Předpokládalo se, že u Wi-Fi komponenty bude použita TCP i UDP komunikace. UDP komunikace byla brána, jako sekundární, protože má sloužit v šíření a příjmu informací o serveru. Za to TCP komunikace bude používána pro výměnu systémových, chatových a aplikačních zpráv.

V OS Android se používá pro TCP komunikaci balík Sockets. V počáteční fázi byl vytvořen jednoduchý server a klient. Kdy server naslouchal na zadaném portu a klient se k němu na známé IP adrese a portu připojil a odeslal mu zprávu, na kterou mu server odpověděl. Následně byla komunikace ukončena. Vše probíhalo výpisem zpráv do LogCat-u. Aby komunikace mohla proběhnout vícekrát, do kódu serveru a klienta byl přidán cyklus, který by prováděl výměnu dat cyklicky.

Aby mohl server komunikovat s více klienty najednou, bylo nutné najít řešení, které bylo nalezeno v použití více vláken. Server má spuštěné vlákno, ve kterém v cyklu čeká na příchozí klienty. Po přijmutí klienta dojde k vytvoření a spuštění nového vlákna, v němž probíhá komunikace s tímto klientem. Objekt spuštěného klientského vlákna je uložen v datové struktuře ArrayList.



Obrázek č. 10 - Vytvoření vlákna pro každého připojeného klienta

Po zprovoznění TCP komunikace s více klienty, následovala implementace multicastu, který používá UDP komunikaci. Hlavním úkolem je v tomto případě usnadnit klientovi vyhledávání a připojení k serveru. Multicast je v OS Android realizován pomocí MulticastSockets. Každý MulticastSocket vysílá a přijímá data na zadaném portu, který je ve Wi-Fi komponentě nastaven na 4444. Také je velice důležité přidání MulticastSocketu do skupiny, která je realizována multicastovou IP adresou. V tomto případě 224.2.76.24. Klient i server se musí nacházet ve stejné skupině. Přiřazení do skupiny se provádí metodou *joinGroup()*. Po nastavení MulticastSockets na klientovi a serveru bylo nutné zabezpečit, jaká data se mají přenášet. Byl vytvořen datagram, který představuje datový paket v protokolu IP, do kterého se vloží potřebná přenášená data. Pro naše potřeby je potřeba název serveru, multicastová IP adresa a port. Klient zaregistrovaný ve stejné multicast skupině jako server, zpracuje přijatý datagram a přečte data.

7.2 Implementace klienta a serveru pro Bluetooth komponentu

Implementace klienta a serveru pro Bluetooth komponentu měla podobný průběh, avšak odlišnou realizaci. A to především lišícím se API v OS Android pro práci s těmito technologiemi. Rovněž bylo vycházeno z předem navrženého UML diagramu (finální UML diagram se nachází na obrázku č. 6).

Aby mohly být prováděny žádosti o připojení, přenos dat, vyhledávání nových zařízení a manipulaci s nastavením Bluetooth, je nutné přidání oprávnění do souboru AndroidManifest.xml. Oprávnění, která byla přidána do toho souboru, jsou: *android.permission.BLUETOOTH_ADMIN* a *android.permission.BLUETOOTH*.

Zapínání Bluetooth v aplikacích není možné provádět bez souhlasu uživatele, jako je tomu u Wi-Fi. Je nutné spustit systémovou aktivitu *BluetoothAdapter.ACTION_REQUEST_ENABLE*, která vyvolá dialog s dotazem na uživatele, zda-li umožní povolení spuštění Bluetooth, či nikoli. Totéž platí pro zapnutí viditelnosti Bluetooth pro ostatní zařízení. Jen s rozdílnou systémovou aktivitou *BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE* , u které lze nastavit i čas viditelnosti Bluetooth.

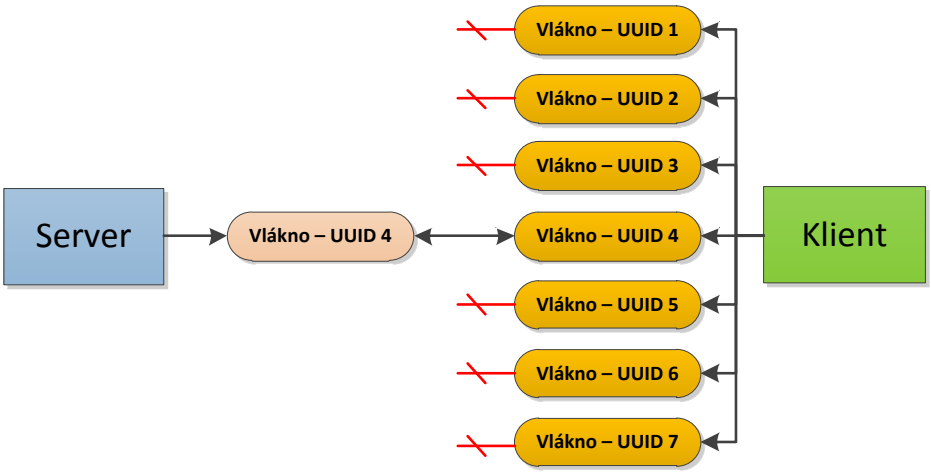
Když byl Bluetooth zapnutý a viditelný pro ostatní zařízení bylo nutné před samotným zajištěním spojení provést tyto akce:

- Spuštění vyhledávání zařízení,
- Zobrazení viditelných nebo již spárovaných zařízení.

Vyhledávání viditelných zařízení se provádí voláním metody startDiscovery(). Kromě toho byla potřebná implementace BroadcastReceiveru, který se přijímá Intents, pokud je nalezeno nové zařízení nebo pokud vyhledávání skončí. Spárovaná zařízení jsou uložena v systému a k jejich získání slouží metoda getBondedDevices(). Následně bylo vytvořeno jednoduché GUI s komponentou ListView, ve které byla zobrazena viditelná a párovaná zařízení s jejich MAC adresy. Pro ListView byl napsán OnItemClickListener, který kliknutím na vybranou položku, která představuje zařízení, vyjme MAC adresu zvoleného zařízení, s nímž bude provedeno spojení.

V první fázi spojení došlo k implementaci jednoduchého klienta a serveru, který přijal pouze jednoho klienta. Jak již bylo v návrhu zmíněno, klientovi nestačí znát pouze MAC adresu serveru, ale musí mít rovněž stejné UUID, aby došlo ke spojení. Server poslouchá na zabezpečeném RFCOMM realizovaným Bluetooth socketem s daným UUID, na příchozího klienta. Klient se naopak snaží připojit k serveru pomocí MAC adresy a SDP identifikátoru UUID. Když byla zařízení spojena, došlo k výměně zpráv, které byly vypsány do systémového logu LogCat.

K realizaci spojení serveru s více klienty, bylo nutné použít více UUID, jelikož každé spojení musí mít své vlastní UUID. Kdyby server poslouchal pouze na jednom UUID, každý nově příchozí klient by se sice připojil, ale došlo by i k odpojení připojeného klienta. Proto má server i klient (z technologických omezení Bluetooth maximálního počtu slave zařízení připojených k master zařízení) 7 jedinečných SPD identifikátoru UUID, které jsou stejné na serveru i klientovi. Server v cyklu naslouchá na volném UUID. Když se klient připojuje k serveru, dojde k spuštění sedmi spojovacích vláken, které se snaží připojit k serveru (Obrázek č. 11). Každé z těchto vláken používá odlišné UUID a pokud je server v dosahu, dojde k spojení na UUID na němž server právě naslouchal. Server uloží objekty připojených klientských vláken do datové struktury ArrayList, která mu umožní s nimi dále pracovat.



Obrázek č. 11 – Bluetooth - připojení klienta k serveru

7.3 Implementace UI

Při implementaci UI bylo postupováno podle předem zvoleného návrhu, ve kterém se počítá s použitím standardních UI komponent, které OS Android nabízí.

Pro každou aktivitu byl vytvořen layout, který odpovídá práci v této aktivitě. Jako základ všech layouts byl zvolen typ Relative, do kterého jsou podle potřeby vkládány další UI komponenty nebo layouts.

V aktivitě, kde si uživatel může zvolit, zda chce být server nebo klient, je použit Relative Layout s dvěma tlačítky (Buttons). Aktivity nabízející nastavení serveru a klienta používají layouts, které používají typ Relative Layout. V něm se nachází TextViews, EditTexts, Button a ScrollView.

V layouts, které používají aktivity k zobrazení seznamu dostupných serveru na klientovi, nebo k zobrazení seznamu připojených klientů na serveru, je použit typ layout Relative. Do něho byly vkládány nejen UI komponenty jako TextViews a Buttons, ale rovněž i další layouts typu Linear a ListView. U Bluetooth komponenty je navíc v těchto layouts přidán Spinner, který představuje filtr zobrazovaného seznamu zařízení. Jestli mají být zobrazeny připojené, viditelné, párované nebo server zařízení.

K informování uživatele pomocí zpráv zobrazovaných na displeji byly implementovány dva typy upozornění AlertDialog a Toast.
Při zobrazení AlertDialogu vyskočí na displeji malé okno, které vyzve uživatele k rozhodnutí nebo zadání dalších informací.

Toast poskytuje jednoduchou odezvu o probíhajících operacích v malém vyskakovacím oknu. V rámci obou komponent je Toast použit k informování např. stav připojování, stav serveru. AlertDialog je použit k informování, např. když uživatel vyplní potřebné údaje nebo není připojen k Wi-Fi síti.

7.4 Implementace Service a Broadcast

Aby mohlo mít více aktivit přístup k již spuštěným vláknům, které nejsou typu deamon nebo nepoužívají návrhový vzor singelton, byla implementována Bound Service.

Wi-Fi i Bluetooth komponenty mají naimplementovanou svou Bound Service. Třída použita jako Bound Service dědí z třídy Service, která umožňuje další vázání a komunikací aplikace s ní.

K zajištění vazby na Service, musí třída implementovat onBind() metodu, která vrátí IBinder objekt. Ten definuje programové rozhraní, které mohou klienti využít pro komunikaci se Service. Objekt IBinder je vytvořen jako singelton, což znamená, že v rámci programu je vytvořena pouze jedna instance třídy. V těle třídy Bound Service jsou vytvořeny metody pro

práci s jednotlivými vlákny. Je zde také vytvořen Handler, který umožňuje vláknům posílání Toasts zpráv právě zobrazeným aktivitám.

Klient se naváže k Service voláním metody bindService(). Zároveň, musí poskytovat implementaci ServiceConnection, která monitoruje spojení se Service. Metoda bindService() se okamžitě vrátí bez hodnoty, ale když OS Android vytváří spojení mezi klientem a Service, volá metodu onServiceConnected() v ServiceConnection, která dodá IBinder, který klient používá pro komunikaci se Service.

Klienty jsou v tomto případě aktivity, které volají Service metodu bindService() v metodě onStart(). Zrušení spojení mezi klientem a Service provádí metoda unbindService(). Pokud se od Service odpojí všichni klienti, dojde k zastavení Service.

K zajištění komunikace vláken s aktivitami, byl implementován BroadcastReceiver do některých aktivit, které potřebují mít od těchto vláken zpětnou vazbu.
Aktivity mají zaregistrované potřebné receivers, na které reaguje BroadcastReceiver. Service předává Context aktivit komunikačním vláknům, které díky tomu mohou posílat Broadcast s požadovanými Intents akcemi, na které BroadcastReceiver zareaguje, pokud je má aktivita zaregistrované.

7.5 Implementace komunikačního protokolu

Komunikační protokol pro obě komponenty má být co nejvíc variabilní a rozšiřitelný, aby umožňoval vývojáři přizpůsobit komunikaci podle svých představ. Byly sice navrženy 3 typy zpráv, které však nemusí být používány. Jak již bylo zmíněno v návrhu, formátem pro výměnu dat byl zvolen JSON. Pro práci s formátem JSON se v OS Android používá balík *org.json*.

Nejdříve byly testovací data vloženy do JSON objektu pomocí metody *put (název, hodnota)* a následně převedeny na String, který byl odeslán. Tento String byl přijat a vypsán do LogCat-u, kde byl sice čitelný, ale nebyl rozparsován. To znamená, že z něj nebyly získány potřebné informace.
Aby byly tyto informace získány, byl napsán jednoduchý parser, který vytvoří nový JSONObject s názvem/hodnotou pomocí mapování, z přijatého String-u. Z takto vytvořeného objektu se získávají informace metodami *get* např. getInt(),getDouble(),getString() a dalšími. Následně se uloží do patřičných proměnných. Když bylo jednoduché zapisování do JSON objektu, také odeslání, přijetí a následné parsování funkční, došlo k implementaci předem navržených zpráv.

K variabilitě a rozšiřitelnosti byl pro každý typ zprávy naimplementován vlastní parser a „zapisovač“ do JSON objektu. Každý z nich je realizován novou třidou, a to z důvodu lepší orientace. Tím je umožněno vývojáři vytváření nových zpráv nebo úprava již

naimplementovaných zpráv. Každý typ zprávy má svůj číselný identifikátor. Když je zpráva odesílána, musí být zadán číselný identifikátor zprávy, aby mohl být vybrán správný „zapisovač“ do JSON objektu. Naopak, když je zpráva přijatá, dojde k zjištění číselného identifikátoru zprávy a naimplementovaný příkaz switch, podle něhož zavolá potřebný parser.

7.6 Problémy při implementaci

Během implementace Wi-Fi a Bluetooth komponenty, se objevily nejrůznější problémy, které bylo potřeba vyřešit. Některé z těchto problémů byly závažnější a jejich řešení zabralo více času, některé byly méně závažné. V následujícím textu budou uvedeny některé z těchto problému a jejich řešení.

Bluetooth komponenta - Opakované připojení klientů

Když bylo realizováno spojení s jedním nebo více klienty nastal problém. Problém se objevil ve chvíli, kdy se klient odpojil a následně se chtěl opět připojit. Klient hlásil, že se k serveru připojil, ale nemohl přijímat a odesílat data. Server nedetekoval opět připojeného klienta a rovněž s ním nemohl komunikovat. Problém byl vyřešen zavíráním BluetoothServerSocket po každém přijatém klientovi a jeho opětovném otevření. Toto řešení je specifické pro BlutoothServerSocket, u „klasického“ ServerSocketu to není nutné. Problém způsoboval BluetoothServerSocket, který současně naslouchal na více SDP identifikátorech UUID, jelikož nedošlo k jejich zavření po přijetí klienta.

Bluetooth komponenta - Vyhledání serveru

Bluetooth umožňuje vyhledání všech viditelných zařízení v dosahu nebo zobrazení všech spárovaných zařízení, ale není možné zjistit, které z nich je server. Šíření informací o serveru Broadcast nebo Multicast kanálem Bluetooth neumožňuje. Bylo potřeba najít jiné řešení. Tento problém byl vyřešen přejmenováním jména Bluetooth adapteru na straně serveru. Bluetooth adapter se přejmenuje na jméno serveru, které si zvolí uživatel. Před toto jméno je vložen specifický String (v tomto případě server-). Když pak klient vyhledává viditelné zařízení, jsou zobrazeny pouze ty, které začínají na tento specifický String. Původní jméno Bluetooth adapteru je uloženo, a když dojde k ukončení Service je mu zpět vráceno.

Wi-Fi komponenta - Posílání informací o serveru přes MulticastSocket

Při zasílaní informací o serveru pomocí MulticastSocket, se objevil další problém. Když byly tyto informace zasílány cyklicky na AP bez časové prodlevy, docházelo k přetížení AP. Toto přetížení způsobovalo odpojování od AP a restartování AP. Problém byl vyřešen vložením časové prodlevy mezi odesíláním jednotlivých datagramů. Konkrétně uspáním vlákna pomocí metody *sleep(),* na jednu vteřinu.

Wi-Fi komponenta - Klient zobrazoval servery, které již neexistují.

Pokud klient obdrží informace o serveru, které jsou následně zobrazeny v seznamu a server ukončí svou činnost, klientovi se tento server neodebere ze seznamu. K vyřešení toho problému přispělo, že do každého přijatého a uloženého datagramu na straně klienta, byl vložen systémový čas. Tento čas je aktualizován pouze příchozími datagramy od serveru, který je v seznamu. Při spuštění vyhledávání serveru, je spuštěn i časovač, který v určitých časových intervalech spustí úlohu, která porovná, jestli rozdíl aktuálního času a času uloženého v datagramu není větší, než je povoleno. Pokud ano, je uložený datagram smazán a UI provede obnovení seznamu.

Wi-Fi komponenta - Připojení k serveru z hlavního vlákna

Když se chtěl klient připojit k serveru a spouštěl komunikační vlákno v aktivitě, došlo k pádu aplikace a byla vyvolána výjimka *NetworkOnMainThreadException*. Tato výjimka se nachází v API OS Android od API 11 (Android 3.0), na nižších verzích API není k dispozici. Výjimka je volána, když se aplikace pokusí provést síťovou operaci z hlavního vlákna.

Problém byl vyřešen implementací UI vlákna. Pokud je spuštěno, spustí síťovou operaci, která v mém případě představuje komunikační vlákno.

Service - Zabránění režimu spánku

Bound Service je spuštěná, když je k ní přihlášen alespoň jeden klient (aktivita). Pokud je zařízení uspáno, dojde nejen k volání metody onPause(), ale i k volání onStop(). Následnému zrušení spojení mezi klientem a Service. Při zrušení spojení, není-li připojen další klient, je Service zrušena, což vede v tomto případě k zastavení některých vláken.

K vyřešení tohoto problému, je v potřebné aktivitě volána v metodě onCreate(), metoda getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON), která přidá k dané obrazovce parametr *FLAG_KEEP_SCREEN_ON*. Díky tomu je zabráněno vypnutí displeje, po dobu viditelnosti daného okna.

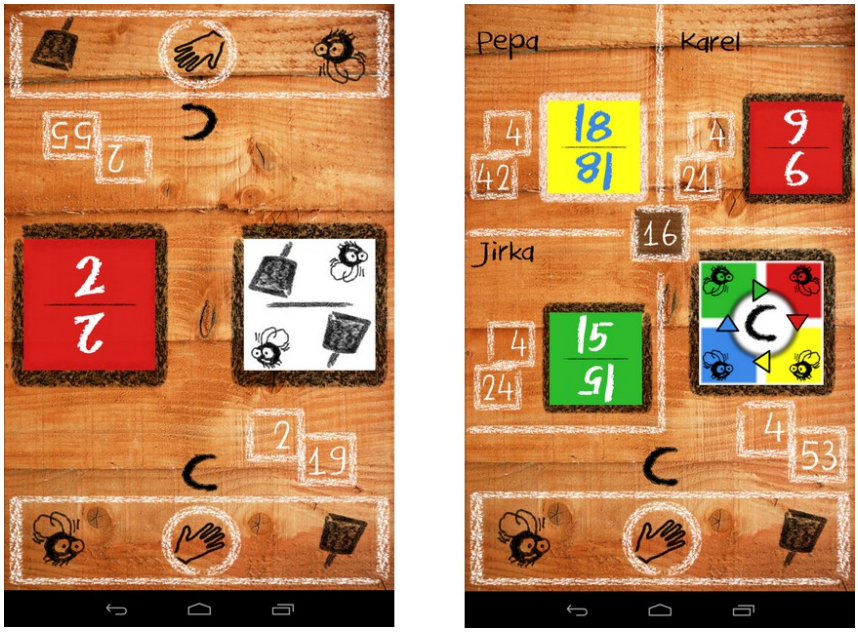
8. Otestování obou komponent v reálné aplikaci

Poté co byly obě komponenty naimplementovány, zbývalo ověření jejich funkčnosti v reálné aplikaci. Pro tento účel byla vytvořena karetní hra SwatFly, která vychází z deskové hry Jungle Speed. Tato desková hra je určena pro dva až osm hráčů. Před samotnou implementací bylo nutné nastudování pravidel deskové hry, aby se tato hra dala převést do virtuální podoby. Bylo nutné přizpůsobení hry a jejich pravidel a nahrazení herních prvků a principů.

Než byla realizována hra pro více hráčů na více zařízeních, byla nejprve vytvořena hra pro dva hráče na jednom zařízení. Nejdříve byly naimplementovány herní pravidla a následně byla navržena a naimplementována grafická podoba hry, která byla optimalizována na všechny rozlišení a DPI displeje.

V momentě když hra byla pro dva hráče na jednom zařízení funkční, došlo k rozšíření hry o multiplayer, realizovaný již naimplementovanými Wi-Fi a Bluetooth komponenty.

Do rozpracované hry byly naimportovány všechny balíky komponent. V aplikaci byla nastavena, v souboru AndroidManifest.xml, potřebná oprávnění, které vyžadují komponenty pro práci s Wi-Fi a Bluetooth. Do téhož souboru byly přidány aktivity z komponent, se kterými bude pracováno. Nejprve došlo k otestování spojení mezi více zařízeními a ověření správné reakce na systémové zprávy.



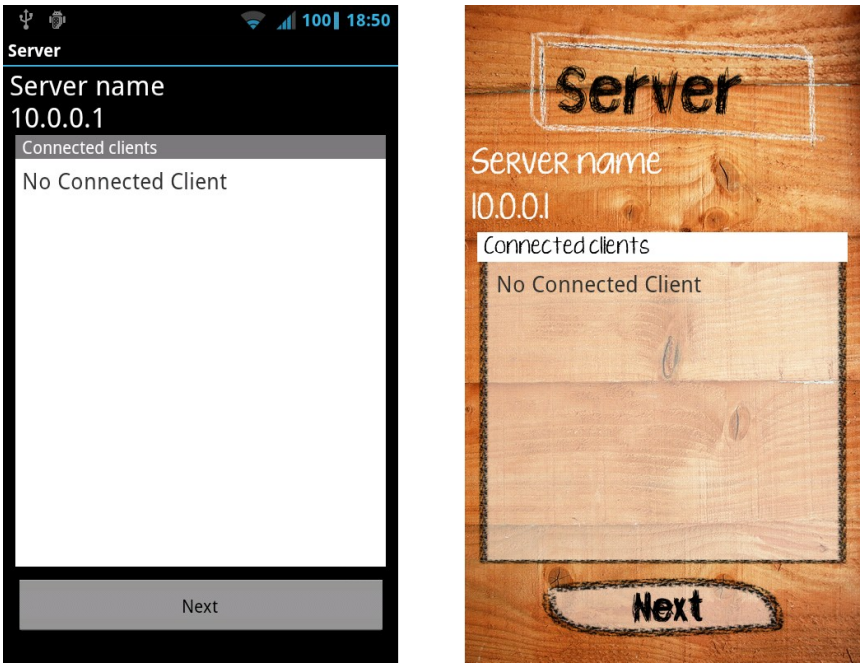
Obrázek č. 12 - UI hry, vlevo- pro2 hráče jednom zařízení, vpravo - pro 4 hráče na 4 zařízeních

Nejdůležitějším krokem bylo navržení a realizace aplikačních zpráv. Především však jejich struktury. Následně byl naimplementován „zapisovač“ do JSON objektu a parser pro aplikační typy zpráv.

Samozřejmě nedošlo okamžitě ke kompletní realizaci těchto zpráv. Aplikační zprávy byly nejprve rozděleny na jednotlivé části, které byly realizovány a testovány. Když byly všechny části zpráv funkční, došlo k jejich sloučení.

Při návrhu UI pro hru více hráčů, bylo nutné předpokládat, že s přibývajícím počtem hráčů, poroste i počet zobrazovaných prvků. Aby byla zachována čitelnost a přehlednost UI hry pro více hráčů i na menších displejích, byl omezen maximální počet hráčů na čtyři. Protože hru nebudou hrát vždy čtyři hráči, ale i dva nebo tři, byly navrženy a naimplementovány tři layouty, které se podle připojeného počtu hráčů nastaví na displeji. Také bylo upraveno UI komponent (Obrázek č. 13), aby bylo sjednoceno UI hry.

Během celého vývoje, byla hra testována a tím mohly být odhaleny a následně odstraněny její nedostatky nebo chyby. Poté co byla hra odladěna, došlo k umístění hry na online distribuční službu Google Play odkud jsi si ji může kdokoliv, vlastnící zařízení v OS Android, stáhnout.



Obrázek č. 13 - Porovnání UI serveru, vlevo- defaultní UI komponenty, vpravo - upravené UI hry

9. Závěr

Cílem této diplomové práce bylo zaměřit se na vývoj aplikací pro mobilní telefony Android. Konkrétně vyvinutí komponent Wi-Fi ad-hoc sítě a Bluetooth sítě, které měly poskytnout dalším vývojářům základ ke komunikaci a hraní na více zařízeních přes Bluetooth nebo v lokální síti přes Wi-Fi pro jejich aplikace nebo hry. Nejprve byla provedena rešerše současného stavu používaných modulů Wi-Fi a Bluetooth v zařízeních s OS Android. Bylo zjištěno, že se v zařízeních s OS Android nalézají Bluetooth a Wi-Fi v jednom čipu, ve kterém může být i FM rádio nebo GPS. Nejpoužívanějším výrobcem těchto čipů v zařízeních s OS Android je společnost Broadcom. Existují rovněž výrobci modulů, kteří poskytují ucelenější řešení.

Poté proběhl návrh, který byl rozdělen do pěti částí, z nichž se každá zabývá specifickými stavebními bloky komponent. Podle zvoleného návrhu byla provedena implementace obou komponent, při které se bylo nutné vypořádat s některými problémy. V poslední fázi proběhla realizace obou komponent. Byla vytvořena hra, která využívá možnosti nabízené v těchto komponentách.

Hlavním přínosem této práce je vytvoření komponent, které budou sloužit dalším vývojářům. Umožní jim do jejich nové nebo stávající aplikace či hry snáze implementovat komunikaci a hraní na více zařízeních přes Bluetooth nebo v lokální síti přes Wi-Fi.

Nebudou muset implementovat své vlastní řešení, ale využijí již nabízené komponenty. Umožní jim vytváření serveru a klienta. V roli serveru nabídne vytvoření nového serveru, šíření informací o sobě samém přes multicast (pouze o Wi-Fi komponenty) a počet a stav připojených klientů. V roli klienta nabídne vyhledání všech dostupných serverů, připojení a komunikaci k serveru. Komponenty poskytují vývojářům základní UI, které si můžou přizpůsobit svým potřebám nebo si v případě potřeby můžou vytvořit své vlastní UI. Vývojářům usnadňují komponenty vytváření komunikačního protokolu pro jejich aplikace nebo hry, již vytvořenými systémovými a chatovými typy zpráv.

Tyto komponenty nejsou určeny pro jednu konkrétní aplikaci nebo hru. Je plánováno jejich nasazení do dalších aplikací a her. Komponenty jsou uvolněny jako Open Source, takže je budou moci využívat zdarma i další vývojáři. Komponenty budou dále vyvíjeny a rozšiřovány.

Seznam použité literatury

[1] LEE, Wei-Meng. Beginning Android application development. Indianapolis, IN: Wiley Pub., c2011, xx, 428 p. Wrox beginning guides. ISBN 978-111-8087-800.

[2] ZECHNER, Mario. Beginning Android games. New York: Apress, c2011, xvi, 669 s. Wrox beginning guides. ISBN 978-1-4302-3042-7.

[3] GARGENTA, Marko. Learning Android. 1st ed. Sebastopol, Calif.: O'Reilly, c2011, xvii, 245 p. ISBN 14-493-9050-1.

[4] *Android – App – Market.com* [online]. c2012 [cit. 03-01-2013]
Dostupné z www: < http://www.android-app-market.com/android-architecture.html>

[5] *Android Developers* [online]. [cit. 15-01-2013]
Dostupné z www: < http://developer.android.com/guide/components/fundamentals.html>

[6] *Android Blog* [online]. c2012 [cit. 15-01-2013]
Dostupné z www: < http://android.programmerguru.com/activity-life-cycle/# >

[7] *Xamarin* [online]. c2013 [cit. 15-01-2013]
Dostupné z www:
< http://docs.xamarin.com/guides/android/application_fundamentals/activity_lifecycle >

[8] *Bluetooth SPECIAL INTEREST GROUP* [online]. c2013 [cit. 20-01-2013]
Dostupné z www: < http://www.bluetooth.org/ >

[9] *Bluetooth Developer Portal* [online]. c2013 [cit. 20-01-2013]
Dostupné z www: < http://developer.bluetooth.org/ >

[10] BY H. LABIOD, By H.H. Wi-Fi, Bluetooth, Zigbee and WiMAX. Dordrecht: Springer, 2007. ISBN 90-481-7359-0.

[11] BY H. LABIOD, By H.H. Bluetooth Low Energy The Developer's Handbook. Dordrecht: Prentice Hall, 2007. ISBN 01-328-8836-X.

[12] HARTE, Lawrence. Introduction to Bluetooth: technology, market, operation, profiles, and services. 2nd ed. Fuquay-Varina, NC: Althos Pub, 2010. ISBN 19-328-1372-1.

[13] Palo wireless Bluetooth Resource Center [online]. c2002 [cit. 10-02-2013]
Dostupné z www: < http://www.palowireless.com/bluearticles/cc5_newprofiles.asp# >

[14] WWW.EDITUJ.CZ, *AUTOMA časopis pro automatizační techniku* [online]. c2013 [cit. 15-02-2013] Dostupné z www: <
http://www.odbornecasopisy.cz/index.php?id_document=28874 >

[15] *WEBOPEDIA* [online]. c2013 [cit. 15-02-2013]
Dostupné z www: < http://www.webopedia.com/TERM/W/Wi_Fi.html >

[16] *Wi-Fi Alliance* [online]. c2013 [cit. 16-02-2013]
Dostupné z www: < http://www.wi-fi.org/ >

[17] *About.com Wireless/Networking* [online]. c2013 [cit. 20-02-2013]
Dostupné z www:
<
http://compnetworking.about.com/od/wireless/WiFi_Wireless_Networks_and_Technology.htm
>

[18] *About.com Wireless/Networking* [online]. c2013 [cit. 20-02-2013]
Dostupné z www:
< http://compnetworking.about.com/cs/wirelessfaqs/f/adhocwireless.htm >

[19] Wi-Fi Alliance [online]. c2013 [cit. 25-02-2013]
Dostupné z www: < http://www.wi-fi.org/discover-and-learn/wi-fi-direct >

[20] TREEHOUSE, *CHIP.CZ* [online]. c2011 [cit. 27-02-2013]
Dostupné z www: < http://www.chip.cz/clanky/internet/2011/10/wi-fi-direct-prime-spojeni >

[21] GAST, Matthew. 802.11 wireless networks: the definitive guide. 2nd ed. Sebastopol: O´Reilly, 2005, xxi, 630 s. ISBN 978-0-596-10052-0.

[22] *About.com Wireless/Networking* [online]. c2013 [cit. 05-03-2013]
Dostupné z www:
< http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm >

[23] *Kioskea.net* [online]. c2013 [cit. 05-03-2013]
Dostupné z www: < http://en.kioskea.net/contents/802-introduction-to-wi-fi-802-11-or-wifi >

[24] *Úvod do JSON* [online]. [cit. 06-03-2013]
Dostupné z www: < http://www.json.org/json-cz.html >

[25] *Zdroják.cz* [online]. c2009 [cit. 07-03-2013]
Dostupné z www: < http://www.zdrojak.cz/clanky/json-na-nekolik-zpusobu/ >

[26] *MOZILLA DEVELOPER NETWORK* [online]. c2012, poslední revize 9.8.2012 [cit. 07-03-2013]
Dostupné z www: < https://developer.mozilla.org/en-US/docs/JSON >

[27] DOZUKI. *Ifixit* [online]. c2013 [cit. 10-03-2013]
Dostupné z www: < http://www.ifixit.com/ >

[28] *Chipworks* [online]. c2013 [cit. 10-03-2013]
Dostupné z www: < http://www.chipworks.com/ >

[29] *TechRepublic* [online]. c2013 [cit. 10-03-2013]
Dostupné z www: < http://www.techrepublic.com/ >

[30] *BROADCOM*. [online]. c2013 [cit. 11-03-2013]
Dostupné z www: < http://www.broadcom.com/ >

[31] *Qualcomm* [online]. c2012 [cit. 15-03-2013]
Dostupné z www: < http://www.qualcomm.com/ >

[32] *muRata* [online]. c2013 [cit. 17-03-2013]
Dostupné z www: < http://www.murata.com/ >

[33] *AzureWave* [online]. c2013 [cit. 18-03-2013]
Dostupné z www: < http://www.azurewave.com/ >

[34] *Texas Instrument* [online]. c2013 [cit. 20-03-2013]
Dostupné z www: < http://www.ti.com/ >

[35] *innovantes* [online]. c2011 [cit. 01-04-2013]
Dostupné z www: < http://www.innovantesindia.com/wordpress/2011/04/01/wifi/ >

[36] WORDPRESS.- STARDUST*, AndEngine* [online]. c2013 [cit. 05-04-2013]
Dostupné z www: < http://www.andengine.org/blog/ >

[37] *mages* [online]. c2010 [cit. 05-04-2013]
Dostupné z www: < http://code.google.com/p/mages/ >

[38] *RakNet* [online]. [cit. 06-04-2013]
Dostupné z www: < http://www.jenkinssoftware.com/ >

[39] *Photon server* [online]. c2013 [cit. 07-04-2013]
Dostupné z www: < http://www.exitgames.com/ >

[40] *Skiller* [online]. c2012 [cit. 06-04-2013]
Dostupné z www: < http://www.skiller-games.com/ >

[41] *IDC* [online]. c2013 [cit. 07-04-2013]
Dostupné z www: < http://www.idc.com/getdoc.jsp?containerId=prUS23946013 >

A. Programátorská příručka Wi-Fi a Bluetooth komponenty

Obě komponenty poskytují základ ke komunikaci a hraní na více zařízeních přes Bluetooth nebo v lokální síti přes Wi-Fi pro jejich aplikace nebo hry.

Wi-Fi komponenta:

- Obsahuje 3 balíky
  - com.sf.wifi_component
  - com.sf.wifi_gui
  - com.sf.json
- Od Android API 4

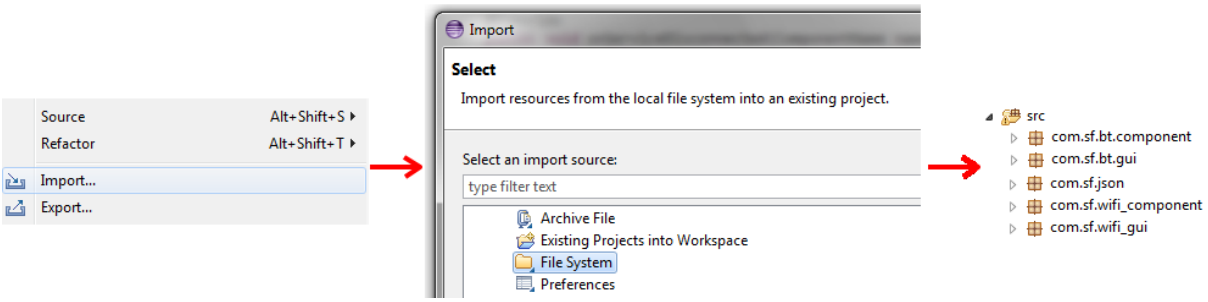
Bluetooth komponenta:

- Obsahuje 3 balíky
  - com.sf.bt.component
  - com.sf.bt.gui
  - com.sf.json
- Od Android API 5

Přičemž balík com.sf.json mají obě komponenty totožné.

1. Import balíku do projektu

Ve vývojovém prostředí se vloží do Android projektu balíky vybrané komponenty pomocí *Import/File Systém/cesta k souborům* (Obrázek A. 1). Na Obrázku A. 1 vpravo, jsou importovány obě komponenty



Obrázek A. 1. - Import ve vývojovém prostředí

2. Oprávnění

K tomu, aby mohly komponenty využívat Wi-Fi nebo Bluetooth je jím nutné povolit určitá oprávnění do souboru manifest.

Bluetooth - oprávnění

Při použití Bluetooth komponenty je potřebné přidat příslušné oprávnění do manifest souboru aplikace. Všechny potřebné oprávnění jsou na obrázku A. 2.

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
  <uses-permission android:name="android.permission.BLUETOOTH" />
```

Obrázek A.2 – Přidání oprávnění pro Bluetooth komponentu do manifest souboru

Wi-Fi - oprávnění

Při použití Wi-Fi komponenty je potřebné přidat příslušné oprávnění do manifest souboru aplikace. Všechny potřebné oprávnění jsou na obrázku A. 3.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

Obrázek A.3 – Přidání oprávnění pro Wi-Fi komponentu do manifest souboru

3. Přidání aktivit

Jak Wi-Fi, tak Bluetooth komponenta obsahuje své aktivity. V balíku *com.sf.bt.gui* se nachází aktivity Bluetooth komponenty a balíku *com.sf.wifi_gui* se nachází aktivity Wi-Fi komponenty. Aby aktivity byly přístupné systému, je nutné jejich definování v souboru manifest.

```
<manifest ... >
  <application ... >
      <activity android:name="com.sf.wifi_gui.Activity" />
      ...
  </application ... >
  ...
</manifest >
```

Obrázek A.4 – Ukázka přidání aktivity do manifest souboru

Layouts k pro aktivity

Pro každou aktivitu pro tyto komponenty byl vytvořen základní layout, který má sloužit jako základ pro další úpravy. Layouts se nachází ve složce *Components/typ komponenty/res/layout*, které je rovněž nutné importovat do Android projektu.

4. Herní (aplikační) aktivita

V obou komponentách se nachází aktivita *GameAc*t, která slouží jako vzor, pro hlavní herní nebo aplikační aktivitu. V následujících krocích bude ukázáno, co musí tato aktivita mít ke své správné funkčnosti implementováno.

a) **Service**
   V první řadě musí být Service, definována v souboru manifest.

```
<service android:name="com.sf.wifi_component.ComponentService‟ />
```

Obrázek. A. 5 - Ukázka přidání servicedo manifest souboru

Poté se musí aktivita připojit k Bound Service pomocí, které bude umožněno odesílání zpráv přes komunikační vlákno a následně je nutné vytvořit třídu *ServiceConnection* pro interakci s hlavním rozhraním Service.

```
ComponentService mService;
private ServiceConnection mConnection = new ServiceConnection() {
    // Vyvolá se po navázání spojení se službou
    public void onServiceConnected(ComponentName className, IBinder
service) {
                LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
           }
    // Vyvolá se při spojení se službou odpojí neočekávaně
    public void onServiceDisconnected(ComponentName className) {

    }
};
```

Obrázek A. 6 - Ukázka vytvoření třídy ServiceConnection

Svázání se Service probíhá v metodě v onStart() a odvázaní od Service probíhá v metodě v onStop().

```
@Override
    protected void onStart() {
        super.onStart();
        // Svázáni z Service
        bindService(new Intent(this, ComponentService.class),
mConnection,
            Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Odvázání od Service
            unbindService(myServiceConnection);
        }
    }
```

Obrázek A. 7 - Ukázka, svázání a odvázaní od Service

b) BroadcastReciever

K tomu, aby aktivita mohla přijímat broadcast zprávy je nutné mít implementovaný BroadcastReceiver a v metodě v *onResume()* mít zaregistrované požadované receivers (přijímače), na konkrétní Intent akce. Odregistrování všech probíhá receivers probíhá v metodě onPause(). V BroadcastReceiver je pak možné na jednotlivé Intent akce patřičně reagovat.

```
 @Override
    protected void onResume() {
        super.onResume();
       // Vytvoření IntentFiltr-u pro akci odpojení klienta
       IntentFilter filter = new
       IntentFilter(ServerCommTh.CLIENT_DISCONNECT);
       // Zaregistrování receiver pro Intent akci odpojení klienta
       this.registerReceiver(mReceiver, filter);
    }

@Override
    protected void onPause() {
        super.onPause();
        // Odregistrování recivers
        this.unregisterReceiver(mReceiver);
        }
    }
```

Obrázek A. 8 - Ukázka, registrování a odregistrování receiver v aktivitě

c) Odesílání zpráv z aktivity

Odesílání zprav je u serveru a klienta odlišné. Klient odesílá zprávy přes Bound Service, která má instanci na komunikační vlákno. Server odesílá zprávy přes singelton instanci třídy ServerData, která má v datové struktuře ArrayList uloženy všechny instance spuštěných komunikačních vláken.
Před samotným odesláním je podle typu posílané zprávy zvolen příslušný „zapisovač“ do JSON objektu, který vrátí String a následně je odeslán.

```
// Získání instance singelton objektu ServerData
 ServerData dataServer = ServerData.getInstance();
// Odeslání systémové zprávy START všem připojeným klientům
dataServer.sendAllClinents(SystemMsg.ID_SYS,SystemMsg.SYS_START)
;
```

Obrázek A. 9. - Ukázka odeslání zprávy ze serveru

```
// Odeslání systémové zprávy READY serveru
mService.sendMsgServer(SystemMsg.ID_SYS,SystemMsg.SYS_READY);
```

Obrázek A. 10. - Ukázka odeslání zprávy z klienta

d) Příjem zpráv

Pokud v komunikačním vlákně dojde k příjmu zprávy, je volána metoda *parseAndWrite()*, která převede zprávu do JSON objektu. Následně je z JSON objektu získáno ID zprávy, které reprezentuje její typ, podle něhož je zvolen příslušný parser, který zapíše data. Dle zjištěného typu zprávy je také aktivitě zaslána broadcast zpráva z konkrétní Intent akcí. Rovněž může být zaslána zpráva Handler-u, který zobrazí Toast s požadovanou zprávou.

```
// Metoda pro výběr parser-u
private void parseAndWrite(String msg) {
    JSONObject jObj;
    try {
        // Vytvoření JSON objektu pomocí mapovaní String msg
        jObj = new JSONObject(msg);
        int id = jObj.getInt(MsgIds.NAME_ID);
        switch (id) {
        // Větev pro systémové zprávy
        case MsgIds.ID_SYS:
                // Volání parser-u pro systémové zprávy
                jParseSys.parseWriteSys(jObj, dataClient);
                //Zavolání metody pro zaslání broadcast
                sendBroadcastMsg(SYS_MSG);
                // Zavolání metody pro zaslání zprávy Handler-u
                                sendHandlerMsg(H_SYS_MSG)
                break;
        default:
                break;
        }
    } catch (JSONException e) {
    }                                                                    }
```

Obrázek A. 11. - Ukázka zvolení parser-u na základně ID zprávy

B. Obsah přiloženého CD

Na přiloženém médiu naleznete následující složky, které obsahují:

Aplikace\Bluetooth komponenta - zkompilovaný soubor BluetoothComponent.apk
Aplikace\Wi-Fi komponenta - zkompilovaný soubor WifiComponet.apk
Javadoc\Bluetooth komponenta - standardně vygenerovaná dokumentace programového kódu Bluetooth komponenta
Javadoc\Wi-Fi komponenta - standardně vygenerovaná dokumentace programového kódu Wi-Fi komponenta
Otestování komponent - URL hry, obsahující obě komponenty
Projekt\Bluetooth komponenta - zkomprimovaný Android projekt Bluetooth komponenty z IDE Eclipse
Projekt\Wi-Fi komponenta - zkomprimovaný Android projekt Wi-Fi komponenty z IDE Eclipse
Text DP - text diplomové práce ve formátu pdf